Smart Contract Security Audit Report

[2021]

Table Of Contents

1 Executive Summary

On 2021.09.22, the SlowMist security team received the Eden team's security audit application for eden-network, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

The SlowMist security team adopts the strategy of "white box lead, black, grey box assists" to conduct a complete security test on the project in the way closest to the real attack.

The test method information:

Test method	Description
Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code modules through the scripting tool, observing the internal running status, mining weaknesses.
White box testing	Based on the open source code, non-open source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc.

The vulnerability severity level information:

Level	Description
Critical	Critical severity vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project team should evaluate and consider whether these vulnerabilities need to be fixed.
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.

Level	Description
Suggestion	There are better practices for coding or architecture.

2 Audit Methodology

The security audit process of SlowMist security team for smart contract includes two steps:

Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using automated analysis tools.

Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

- Reentrancy Vulnerability
- Replay Vulnerability
- Reordering Vulnerability
- Short Address Vulnerability
- Denial of Service Vulnerability
- Transaction Ordering Dependence Vulnerability
- Race Conditions Vulnerability
- Authority Control Vulnerability
- Integer Overflow and Underflow Vulnerability
- TimeStamp Dependence Vulnerability
- Uninitialized Storage Pointers Vulnerability
- Arithmetic Accuracy Deviation Vulnerability
- tx.origin Authentication Vulnerability

- "False top-up" Vulnerability
- Variable Coverage Vulnerability
- Gas Optimization Audit
- Malicious Event Log Audit
- Redundant Fallback Function Audit
- Unsafe External Call Audit
- Explicit Visibility of Functions State Variables Aduit
- Design Logic Audit
- Scoping and Declarations Audit

3 Project Overview

3.1 Project Introduction

https://github.com/eden-network/governance

commit: e7b55f6e3f9c0d3ede7fd8bb39ae4fa7a4f4e79e

3.2 Vulnerability Information

The following is the status of the vulnerabilities found in this audit:

NO	Title	Category	Level	Status
N1	Deflation token compatibility issues	Design Logic Audit	Suggestion	Confirmed
N2	Risk of excessive authority	Authority Control Vulnerability	Low	Confirmed
N3	Payments Contract Deflation token compatibility issues	Design Logic Audit	Suggestion	Confirmed

NO	Title	Category	Level	Status
N4	User voting rights are lost	Others	Medium	Confirming
N5	Out of gas in the loop	Gas Optimization Audit	Suggestion	Confirmed
N6	The new variable is not assigned	Others	Suggestion	Confirmed
N7	Restrictions can be bypassed	Others	Suggestion	Confirmed
N8	Risk of excessive authority	Authority Control Vulnerability	Low	Confirming

4 Code Overview

4.1 Contracts Description

The main network address of the contract is as follows:

The code was not deployed to the mainnet.

4.2 Visibility Description

The SlowMist Security team analyzed the visibility of major contracts during the audit, the result as follows:

DistributorGovernance			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
add	External	Can Modify State	onlyGov
addBatch	External	Can Modify State	onlyGov

DistributorGovernance			
remove	External	Can Modify State	onlyGov
removeBatch	External	Can Modify State	onlyGov
delegate	External	Can Modify State	onlyDelegatorOrProducer
delegateBatch	External	Can Modify State	onlyDelegator
setRewardSchedule	Public	Can Modify State	onlyGov
rewardScheduleEntry	Public	-	-
rewardScheduleEntries	Public	-	-

EdenNetwork			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
slotOwner	Public	-	-
slotDelegate	Public	-	-
slotCost	External	-	-
claimSlot	External	Can Modify State	nonReentrant
claimSlotWithPermit	External	Can Modify State	nonReentrant
slotBalance	Public	-	-
slotForeclosed	Public	-	-
stake	External	Can Modify State	nonReentrant
stakeWithPermit	External	Can Modify State	nonReentrant

EdenNetwork			
unstake	External	Can Modify State	nonReentrant
withdraw	External	Can Modify State	nonReentrant
setSlotDelegate	External	Can Modify State	onlySlotOwner
setTaxRate	External	Can Modify State	onlyAdmin
setAdmin	External	Can Modify State	onlyAdmin
_claimSlot	Internal	Can Modify State	-
_stake	Internal	Can Modify State	-

EdenNetworkManager			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	External	Can Modify State	initializer onlyAdmin
setAdmin	External	Can Modify State	onlyAdmin
setEdenNetworkProxy	External	Can Modify State	onlyAdmin
getProxyImplementation	Public	-	-
getProxyAdmin	Public	-	-
setProxyAdmin	External	Can Modify State	onlyAdmin
upgrade	External	Can Modify State	onlyAdmin
upgradeAndCall	External	Payable	onlyAdmin

EdenNetworkProxy

EdenNetworkProxy			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Payable	ERC1967Proxy
admin	External	Can Modify State	ifAdmin
implementation	External	Can Modify State	ifAdmin
changeAdmin	External	Can Modify State	ifAdmin
upgradeTo	External	Can Modify State	ifAdmin
upgradeToAndCall	External	Payable	ifAdmin
_admin	Internal	-	-
_setAdmin	Private	Can Modify State	-
_beforeFallback	Internal	Can Modify State	-

EdenToken			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
setMetadataManager	External	Can Modify State	-
mint	External	Can Modify State	-
burn	External	Can Modify State	-
updateTokenMetadata	External	Can Modify State	-
approve	External	Can Modify State	-
increaseAllowance	External	Can Modify State	-

EdenToken			
decreaseAllowance	External	Can Modify State	-
permit	External	Can Modify State	-
transfer	External	Can Modify State	-
transferFrom	External	Can Modify State	-
transferWithAuthorization	External	Can Modify State	-
receiveWithAuthorization	External	Can Modify State	-
getDomainSeparator	Public	-	-
_validateSignedData	Internal	-	-
_approve	Internal	Can Modify State	-
_transferTokens	Internal	Can Modify State	-

LockManager			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
getAmountStaked	External	-	-
getStake	Public	-	-
calculateVotingPower	Public	-	-
grantVotingPower	External	Can Modify State	onlyLockers
removeVotingPower	External	Can Modify State	onlyLockers

MerkleDistributor

MerkleDistributor			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC721
setGovernance	External	Can Modify State	onlyAdmins
addUpdaters	External	Can Modify State	onlyAdmins
removeUpdaters	External	Can Modify State	onlyAdmins
setUpdateThreshold	External	Can Modify State	onlyAdmins
claim	External	Can Modify State	-
updateMerkleRoot	External	Can Modify State	onlyUpdaters
slash	External	Can Modify State	onlySlashers
supportsInterface	Public	-	-
tokenURI	Public	-	-
_setGovernance	Private	Can Modify State	-
_setUpdateThreshold	Private	Can Modify State	-
_increaseAccount	Private	Can Modify State	-

Migrator			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
migrate	External	Can Modify State	-
migrateTo	External	Can Modify State	-

Migrator			
migrateWithPermit	External	Can Modify State	-
migrateToWithPermit	External	Can Modify State	-
_migrate	Internal	Can Modify State	-

Payments			
Function Name	Visibility	Mutability	Modifiers
createPayment	External	Can Modify State	-
createPaymentWithPermit	External	Can Modify State	-
allActivePaymentIds	External	-	-
allActivePayments	External	-	-
allActivePaymentBalances	External	-	-
activePaymentIds	External	-	-
allPayments	External	-	-
activePayments	External	-	-
activePaymentBalances	External	-	-
totalTokenBalance	External	-	-
tokenBalance	External	-	-
paymentBalance	Public	-	-
claimableBalance	Public	-	-
claimAllAvailableTokens	External	Can Modify State	nonReentrant

Payments			
claimAvailableTokenAmounts	External	Can Modify State	nonReentrant
stopPayment	External	Can Modify State	nonReentrant
_createPayment	Internal	Can Modify State	-
_claimTokens	Internal	Can Modify State	-

RewardsManager			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
poolLength	External	-	-
add	External	Can Modify State	onlyOwner
set	External	Can Modify State	onlyOwner
rewardsActive	Public	-	-
getMultiplier	Public	-	-
pendingRewardTokens	External	-	-
pendingSushi	External	-	-
massUpdatePools	Public	Can Modify State	-
updatePool	Public	Can Modify State	-
deposit	External	Can Modify State	nonReentrant
depositWithPermit	External	Can Modify State	nonReentrant
withdraw	External	Can Modify State	nonReentrant

RewardsManager			
emergencyWithdraw	External	Can Modify State	nonReentrant
tokenAllow	External	Can Modify State	onlyOwner
rescueTokens	External	Can Modify State	onlyOwner
setRewardsPerBlock	External	Can Modify State	onlyOwner
setSushiToken	External	Can Modify State	onlyOwner
setMasterChef	External	Can Modify State	onlyOwner
setVault	External	Can Modify State	onlyOwner
setLockManager	External	Can Modify State	onlyOwner
changeOwner	External	Can Modify State	onlyOwner
_deposit	Internal	Can Modify State	-
_withdraw	Internal	Can Modify State	-
_distributeRewards	Internal	Can Modify State	-
_safeSushiTransfer	Internal	Can Modify State	-

TokenRegistry			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
setTokenFormula	External	Can Modify State	onlyOwner
removeToken	External	Can Modify State	onlyOwner
changeOwner	External	Can Modify State	onlyOwner

Vault			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
lockTokens	External	Can Modify State	-
lockTokensWithPermit	External	Can Modify State	-
allActiveLockIds	External	-	-
allActiveLocks	External	-	-
allActiveLockBalances	External	-	-
activeLockIds	External	-	-
allLocks	External	-	-
activeLocks	External	-	-
activeLockBalances	External	-	-
totalTokenBalance	External	-	-
tokenBalance	External	-	-
lockBalance	Public	-	-
claimableBalance	Public	-	-
claimAllUnlockedTokens	External	Can Modify State	-
claimUnlockedTokenAmounts	External	Can Modify State	-
extendLock	External	Can Modify State	-
_lockTokens	Internal	Can Modify State	-
_claimTokens	Internal	Can Modify State	-

Vault			
_add16	Internal	-	-

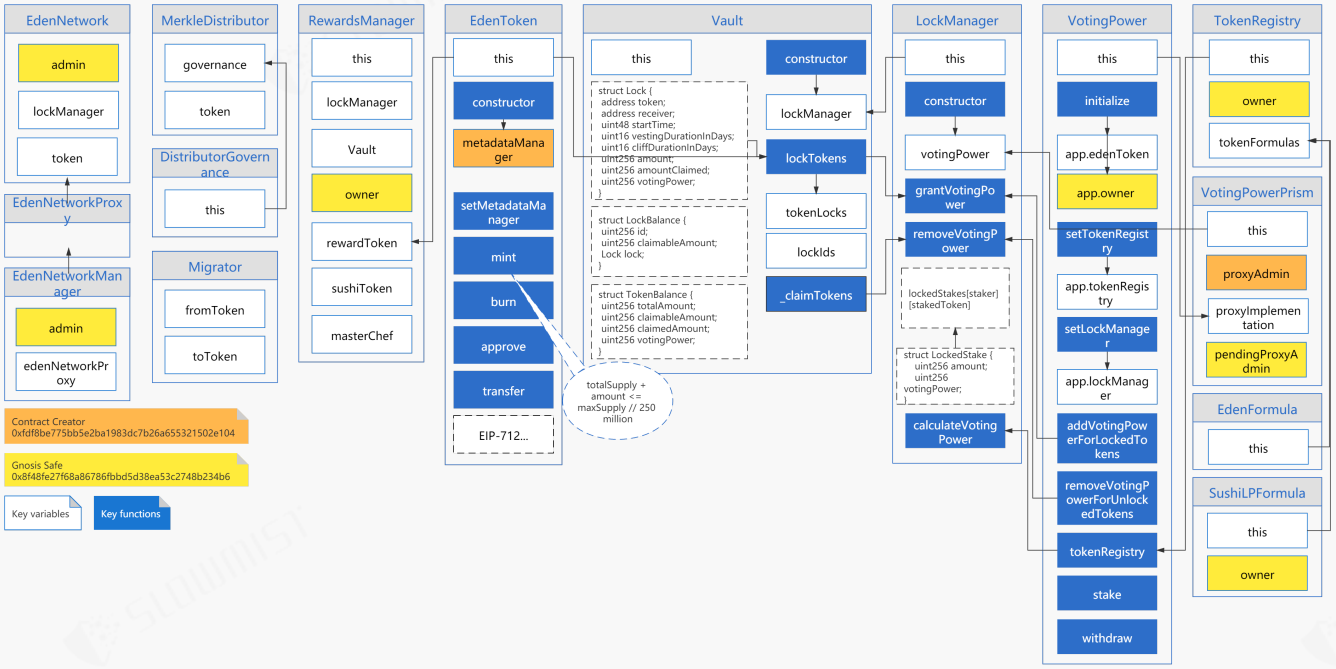
VotingPower			
Function Name	Visibility	Mutability	Modifiers
initialize	Public	Can Modify State	initializer
edenToken	Public	-	-
decimals	Public	-	-
tokenRegistry	Public	-	-
lockManager	Public	-	-
owner	Public	-	-
setTokenRegistry	Public	Can Modify State	onlyOwner
setLockManager	Public	Can Modify State	onlyOwner
changeOwner	External	Can Modify State	onlyOwner
stakeWithPermit	External	Can Modify State	nonReentrant
stake	External	Can Modify State	nonReentrant
stake	External	Can Modify State	nonReentrant
addVotingPowerForLockedTokens	External	Can Modify State	nonReentrant
removeVotingPowerForUnlockedTokens	External	Can Modify State	nonReentrant
withdraw	External	Can Modify State	nonReentrant
withdraw	External	Can Modify State	nonReentrant

VotingPower			
getEDENAmountStaked	Public	-	-
getAmountStaked	Public	-	-
getEDENStake	Public	-	-
getStake	Public	-	-
balanceOf	Public	-	-
balanceOfAt	Public	-	-
_stake	Internal	Can Modify State	-
_withdraw	Internal	Can Modify State	-
_increaseVotingPower	Internal	Can Modify State	-
_decreaseVotingPower	Internal	Can Modify State	-
_writeCheckpoint	Internal	Can Modify State	-
_safe32	Internal	-	-

VotingPowerPrism			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
<Receive Ether>	External	Payable	-
<Fallback>	External	Payable	-

Swimlane diagram

(Does not include contracts/Payments.sol)



4.3 Vulnerability Summary

[N1] [Suggestion] Deflation token compatibility issues

Category: Design Logic Audit

Content

If the number of deflationary token records is smaller than the actual number of receipts, if malicious users continue to `deposit` and `withdraw`, the pool of deflationary tokens will be exhausted.

- contracts/RewardsManager.sol

```
function _deposit(
    uint256 pid,
    uint256 amount,
    PoolInfo storage pool,
    UserInfo storage user
) internal {
    updatePool(pid);

    uint256 sushiPid = sushiPools[address(pool.token)];
    uint256 pendingSushiTokens = 0;
```

```
        if (user.amount > 0) {
            uint256 pendingRewards = user.amount * pool.accRewardsPerShare / 1e12 -
user.rewardTokenDebt;

            if (pendingRewards > 0) {
                _distributeRewards(msg.sender, pendingRewards, pool.vestingPercent,
pool.vestingPeriod, pool.vestingCliff, pool.vpForVesting);
            }

            if (sushiPid != uint256(0)) {
                masterChef.updatePool(sushiPid);
                pendingSushiTokens = user.amount *
masterChef.poolInfo(sushiPid).accSushiPerShare / 1e12 - user.sushiRewardDebt;
            }
        }

        pool.token.safeTransferFrom(msg.sender, address(this), amount);
        pool.totalStaked = pool.totalStaked + amount;
        user.amount = user.amount + amount; //SlowMist Incompatible with deflationary
currencies
        user.rewardTokenDebt = user.amount * pool.accRewardsPerShare / 1e12;

        if (sushiPid != uint256(0)) {
            masterChef.updatePool(sushiPid);
            user.sushiRewardDebt = user.amount *
masterChef.poolInfo(sushiPid).accSushiPerShare / 1e12;
            masterChef.deposit(sushiPid, amount);
        }

        if (amount > 0 && pool.vpForDeposit) {
            lockManager.grantVotingPower(msg.sender, address(pool.token), amount);
        }

        if (pendingSushiTokens > 0) {
            _safeSushiTransfer(msg.sender, pendingSushiTokens);
        }

        emit Deposit(msg.sender, pid, amount);
    }
```

Solution

Check the token balance before and after the recharge as the real recharge amount

Status

Confirmed; The project party will not use deflationary tokens as pledge tokens

[N2] [Low] Risk of excessive authority

Category: Authority Control Vulnerability

Content

These functions may cause the project party to take away the tokens pledged by the user in the pool.

- contracts/RewardsManager.sol

```
    function setMasterChef(address newAddress) external onlyOwner {
        emit ChangedAddress("MASTER_CHEF", address(masterChef), newAddress);
        masterChef = IMasterChef(newAddress);
    }

function setVault(address newAddress) external onlyOwner {
        emit ChangedAddress("VAULT", address(vault), newAddress);
        vault = IVault(newAddress);
    }

    function setLockManager(address newAddress) external onlyOwner {
        emit ChangedAddress("LOCK_MANAGER", address(lockManager), newAddress);
        lockManager = ILockManager(newAddress);
    }

    function rescueTokens(
        address[] calldata tokens,
        uint256[] calldata amounts,
        address receiver
    ) external onlyOwner {
        require(tokens.length == amounts.length, "RM::rescueTokens: not same
length");
        for (uint i = 0; i < tokens.length; i++) {
            IERC20Extended token = IERC20Extended(tokens[i]);
            uint256 withdrawalAmount;
            uint256 tokenBalance = token.balanceOf(address(this));
            uint256 tokenAllowance = token.allowance(address(this), receiver);
            if (amounts[i] == 0) {
                if (tokenBalance > tokenAllowance) {
                    withdrawalAmount = tokenAllowance;
```

```
                } else {
                    withdrawalAmount = tokenBalance;
                }
            } else {
                require(tokenBalance >= amounts[i], "RM::rescueTokens: contract
balance too low");
                require(tokenAllowance >= amounts[i], "RM::rescueTokens: increase
token allowance");
                withdrawalAmount = amounts[i];
            }
            token.safeTransferFrom(address(this), receiver, withdrawalAmount);
        }
    }
```

Solution

It is recommended to transfer the authority to the timelock contract, and to modify the function of the contract parameters using event records.

- The authority related to user funds should be transferred to the timelock contract. The timelock contract admin can use multi-signature to avoid the risk of private key leakage.
- In the early stage of the project, some parameters may be frequently modified (such as: increasing the mortgage pool or reward parameters, etc.). This part of the authority can be controlled separately, and the community is informed about the retention of the authority.
- Consider retaining the authority to temporarily suspend the project in order to respond to an emergency in the early stage of the project, which can quickly suspend the project and stop the loss in time, and at the same time inform the community to retain the authority.
- After the project has passed the early stage of smooth operation, the authority can be transferred to community governance.

Status

Confirmed; After communication, the project party used the 2out of 4 multi-signature address to manage the execution authority!

[N3] [Suggestion] Payments Contract Deflation token compatibility issues

Category: Design Logic Audit

Content

If it is a deflationary currency, the actual number of tokens received at this time is less than the incoming amount.

- contracts/Payments.sol

```
 function _createPayment(
        address token,
        address payer,
        address receiver,
        uint48 startTime,
        uint256 amount,
        uint256 paymentDurationInSecs,
        uint16 cliffDurationInDays
    ) internal {

        // Transfer the tokens under the control of the payment contract
        IERC20(token).safeTransferFrom(payer, address(this), amount);

        uint48 paymentStartTime = startTime == 0 ? uint48(block.timestamp) :
startTime;

        // Create payment
        Payment memory payment = Payment({
            token: token,
            receiver: receiver,
            payer: payer,
            startTime: paymentStartTime,
            stopTime: 0,
            paymentDurationInSecs: paymentDurationInSecs,
            cliffDurationInDays: cliffDurationInDays,
            amount: amount,// SlowMist Incompatible with deflationary currencies
            amountClaimed: 0
        });

        tokenPayments[numPayments] = payment;//SlowMist Incompatible with
deflationary currencies
        paymentIds[receiver].push(numPayments);
        emit PaymentCreated(token, payer, receiver, numPayments, amount,
paymentStartTime, paymentDurationInSecs, cliffDurationInDays);
```

```
        // Increment payment id
        numPayments++;
    }
```

Solution

Check the token balance before and after the recharge as the real recharge amount

Status

Confirmed

[N4] [Medium] User voting rights are lost

Category: Others

Content

This is an externally called function. There may be a risk of `LOCKER_ROLE` removing the voting rights of locked users in batches.

- contracts/Vault.sol

```
    function claimAllUnlockedTokens(uint256[] memory locks) external {
        for (uint i = 0; i < locks.length; i++) {
            uint256 claimableAmount = claimableBalance(locks[i]);
            require(claimableAmount > 0, "Vault::claimAllUnlockedTokens:
claimableAmount is 0");
            _claimTokens(locks[i], claimableAmount);
        }
    }


    function _claimTokens(uint256 lockId, uint256 claimAmount) internal {
        Lock storage lock = tokenLocks[lockId];
        uint256 votingPowerRemoved;

        // Remove voting power, if exists
        if (lock.votingPower > 0) {
            votingPowerRemoved = lockManager.removeVotingPower(lock.receiver,
lock.token, claimAmount);
            lock.votingPower = lock.votingPower - votingPowerRemoved;
```

```
        }

        // Update claimed amount
        lock.amountClaimed = lock.amountClaimed + claimAmount;

        // Release tokens
        IERC20Permit(lock.token).safeTransfer(lock.receiver, claimAmount);
        emit UnlockedTokensClaimed(lock.receiver, lock.token, lockId, claimAmount,
votingPowerRemoved);
    }
```

Solution

`_claimTokens` should be operated by users themselves

Status

Confirming

[N5] [Suggestion] Out of gas in the loop

Category: Gas Optimization Audit

Content

The delegate is an external function. If too many producers are added by malicious calls, it will cause the execution of `delegateBatch` and cause out of gas in the loop.

- contracts/DistributorGovernance.sol

```
    /// @notice Only addresses with delegator role or block producer
    modifier onlyDelegatorOrProducer(address producer) {
        require(hasRole(DELEGATOR_ROLE, msg.sender) || msg.sender == producer, "must
be producer or delegator");
        _;//SlowMist If msg.sender = producer is satisfied, the judgment can be
passed
    }

function delegate(address producer, address collector) external
onlyDelegatorOrProducer(producer) {
        rewardCollector[producer] = collector;
        emit BlockProducerRewardCollectorChanged(producer, collector);
```

```
    }

    function delegateBatch(address[] memory producers, address[] memory collectors)
external onlyDelegator {
        require(producers.length == collectors.length, "length mismatch");
        for(uint i; i< producers.length; i++) {
            rewardCollector[producers[i]] = collectors[i];
            emit BlockProducerRewardCollectorChanged(producers[i], collectors[i]);
        }
    }
```

Solution

You can add a logic to delete the rewardCollector

Status

Confirmed

[N6] [Suggestion] The new variable is not assigned

Category: Others

Content

`newCliffDuration` This variable was not stored at the end.

- contracts/LockManager.sol

```
function extendLock(uint256 lockId, uint16 vestingDaysToAdd, uint16 cliffDaysToAdd)
external {
        Lock storage lock = tokenLocks[lockId];
        require(msg.sender == lock.receiver, "Vault::extendLock: msg.sender must be
receiver");
        uint16 oldVestingDuration = lock.vestingDurationInDays;
        uint16 newVestingDuration = _add16(oldVestingDuration, vestingDaysToAdd,
"Vault::extendLock: vesting max days exceeded");
        uint16 oldCliffDuration = lock.cliffDurationInDays;
        uint16 newCliffDuration = _add16(oldCliffDuration, cliffDaysToAdd,
"Vault::extendLock: cliff max days exceeded");//SlowMist newCliffDuration This
variable was not stored at the end
        require(newCliffDuration <= 10*365, "Vault::extendLock: cliff more than 10
years");
        require(newVestingDuration <= 25*365, "Vault::extendLock: vesting duration
```

```
more than 25 years");
        require(newVestingDuration >= newCliffDuration, "Vault::extendLock: duration
< cliff");
        lock.vestingDurationInDays = newVestingDuration;
        emit LockExtended(lockId, oldVestingDuration, newVestingDuration,
oldCliffDuration, newCliffDuration, lock.startTime);
    }
```

Solution

Status

Confirmed; Communicating with the project party through subsequent versions will fix this problem.

[N7] [Suggestion] Restrictions can be bypassed

Category: Others

Content

When `existingBidAmount` and `existingSlotBalance` are `0`.`bid >= MIN_BID` This condition can be bypassed.

- contracts/EdenNetwork.sol

```
    function _claimSlot(uint8 slot, uint128 bid, address delegate) internal {
        require(delegate != address(0), "cannot delegate to 0 address");
        Bid storage currentBid = slotBid[slot];
        uint128 existingBidAmount = currentBid.bidAmount;
        uint128 existingSlotBalance = slotBalance(slot);
        uint128 taxedBalance = existingBidAmount - existingSlotBalance;
        require((existingSlotBalance == 0 && bid >= MIN_BID) || bid >=
existingBidAmount * 110 / 100, "bid too small");//slowmist

        uint128 bidderLockedBalance = lockedBalance[msg.sender];
        uint128 bidIncrement = currentBid.bidder == msg.sender ? bid -
existingSlotBalance : bid;
        if (bidderLockedBalance > 0) {
            if (bidderLockedBalance >= bidIncrement) {
                lockedBalance[msg.sender] -= bidIncrement;
            } else {
                lockedBalance[msg.sender] = 0;
                token.transferFrom(msg.sender, address(this), bidIncrement -
```

```
bidderLockedBalance);
            }
        } else {
            token.transferFrom(msg.sender, address(this), bidIncrement);
        }

        if (currentBid.bidder != msg.sender) {
            lockedBalance[currentBid.bidder] += existingSlotBalance;
        }

        if (taxedBalance > 0) {
            token.burn(taxedBalance);
        }

        _slotOwner[slot] = msg.sender;
        _slotDelegate[slot] = delegate;

        currentBid.bidder = msg.sender;
        currentBid.periodStart = uint64(block.timestamp);
        currentBid.bidAmount = bid;
        currentBid.taxNumerator = taxNumerator;
        currentBid.taxDenominator = taxDenominator;

        slotExpiration[slot] = uint64(block.timestamp + uint256(taxDenominator) *
86400 / uint256(taxNumerator));

        emit SlotClaimed(slot, msg.sender, delegate, bid, existingBidAmount,
taxNumerator, taxDenominator);
    }
```

Solution

The minimum value of the incoming bid can be judged separately

Status

Confirmed

[N8] [Low] Risk of excessive authority

Category: Authority Control Vulnerability

Content

Since most of the contract settings only need to be modified by the admin, they can take effect immediately. If the admin address is hacked, it will cause some serious consequences.

For example:

- contracts/VotingPowerPrism.sol

  The admin has the authority to modify the pointed logical contract immediately. It may cause abnormal voting rights after the admin address is stolen.

- contracts/EdenNetworkProxy.sol

  The admin has the authority to modify the pointed logical contract immediately. It may cause the loss of the token of the contract address after the admin address is stolen.

- contracts/TokenRegistry.sol

  If the admin address is hacked, you can also modify the TokenFormula immediately. This will cause abnormal voting rights.

Solution

Use the timelock contract to have a time limit for changing permissions. This way, even if there is a problem, the user has time to log out in time.

Status

Confirming

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002109300002	SlowMist Security Team	2021.09.22 - 2021.09.30	Low Risk

Summary conclusion: Summary conclusion: The SlowMist security team use a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 1 medium risk, 1 low risk, 5 suggestion vulnerabilities. And 1 low risk, 5 suggestion vulnerabilities were confirmed and being fixed;

6 Statement

SlowMist issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility based on these.

For the facts that occurred or existed after the issuance, SlowMist is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to SlowMist by the information provider till the date of the insurance report (referred to as "provided information"). SlowMist assumes: The information provided is not missing, tampered with, deleted or concealed. If the information provided is missing, tampered with, deleted, concealed, or inconsistent with the actual situation, the SlowMist shall not be liable for any loss or adverse effect resulting therefrom. SlowMist only conducts the agreed security audit on the security situation of the project and issues this report. SlowMist is not responsible for the background and other conditions of the project.

Official Website

www.slowmist.com

E-mail

team@slowmist.com

Twitter

@SlowMist_Team

Github

https://github.com/slowmist