Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2025.09.28, the SlowMist security team received the Sigma Money team's security audit application for Sigma DAO round 1, 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.
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:

Serial Number	Audit Class	Audit Subclass
1	Overflow Audit	-
2	Reentrancy Attack Audit	-
3	Replay Attack Audit	-
4	Flashloan Attack Audit	-
5	Race Conditions Audit	Reordering Attack Audit
6	Permission Vulnerability Audit	Access Control Audit
		Excessive Authority Audit
7	Security Design Audit	External Module Safe Use Audit
		Compiler Version Security Audit
		Hard-coded Address Security Audit
		Fallback Function Safe Use Audit
		Show Coding Security Audit
		Function Return Value Security Audit
		External Call Function Security Audit

Serial Number	Audit Class	Audit Subclass
7	Security Design Audit	Block data Dependence Security Audit
		tx.origin Authentication Security Audit
8	Denial of Service Audit	-
9	Gas Optimization Audit	-
10	Design Logic Audit	-
11	Variable Coverage Vulnerability Audit	-
12	"False Top-up" Vulnerability Audit	-
13	Scoping and Declarations Audit	-
14	Malicious Event Log Audit	-
15	Arithmetic Accuracy Deviation Audit	-
16	Uninitialized Storage Pointer Audit	-

3 Project Overview

3.1 Project Introduction

Sigma DAO protocol is forked from Shadow Protocol.

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Ignore the return value of the function	Others	Suggestion	Confirming
N2	Missing zero address check	Others	Suggestion	Confirming

NO	Title	Category	Level	Status
N3	The maximum emission limit of the reward has expired	Design Logic Audit	Medium	Confirming
N4	Risk of excessive token minting	Design Logic Audit	Medium	Confirming
N5	Missing event records	Others	Suggestion	Confirming
N6	Risk of excessive authority	Authority Control Vulnerability Audit	Medium	Confirming
N7	Improper variable declaration	Others	Suggestion	Confirming
N8	Risk of excess preallocation amount	Design Logic Audit	Medium	Confirming

4 Code Overview

4.1 Contracts Description

https://github.com/SigmaMoney/dao/tree/feat/bsc

Initial audit version: bd3e1433914ec6cd20a1d72f18aa6ef507ae4f68

Audit Scope:

```
- contracts/SigmaGauge.sol
- contracts/SigmaVesting.sol
- contracts/VeFunderGauge.sol
- contracts/SigmaFeeDistributor.sol
- contracts/factories/SigmaFeeDistributorFactory.sol
- contracts/factories/SigmaGaugeFactory.sol
- contracts/factories/VeFunderGaugeFactory.sol
- contracts/interfaces/ISigmaFeeDistributor.sol
- contracts/interfaces/ISigmaFeeDistributorFactory.sol
- contracts/interfaces/ISigmaGaugeFactory.sol
- contracts/interfaces/ISigmaVesting.sol
- contracts/interfaces/IVeFunderGauge.sol
- contracts/interfaces/IVeFunderGaugeFactory.sol

- contracts/libraries/RewardClaimers.sol
```

```
- contracts/xShadow/XShadow.sol
- contracts/Minter.sol
- contracts/Shadow.sol
- contracts/Voter.sol
- contracts/AccessHub.sol
- contracts/interfaces/IAccessHub.sol
- contracts/interfaces/IMinter.sol
- contracts/interfaces/IVoter.sol
```

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:

SigmaFeeDistributorFactory			
Function Name	Visibility	Mutability	Modifiers
createFeeDistributor	External	Can Modify State	-

SigmaGaugeFactory			
Function Name	Visibility	Mutability	Modifiers
createGauge	External	Can Modify State	-

VeFunderGaugeFactory			
Function Name	Visibility	Mutability	Modifiers
createGauge	External	Can Modify State	-

Gauge			
Function Name	Visibility	Mutability	Modifiers
constructor	Public	Can Modify State	-
rewardsList	External	-	-

Gauge			
rewardsListLength	External	-	-
lastTimeRewardApplicable	Public	-	-
rewardData	External	-	-
earned	Public	-	-
getReward	Public	Can Modify State	updateReward, nonReentrant
getRewardAndExit	Public	Can Modify State	updateReward, nonReentrant
rewardPerToken	Public	-	-
depositAll	External	Can Modify State	-
depositFor	Public	Can Modify State	updateReward, nonReentrant
deposit	Public	Can Modify State	-
withdrawAll	External	Can Modify State	-
withdraw	Public	Can Modify State	updateReward, nonReentrant
left	Public	-	-
whitelistReward	External	Can Modify State	-
removeRewardWhitelist	External	Can Modify State	-
notifyRewardAmount	External	Can Modify State	updateReward, nonReentrant
isWhitelisted	Public	-	-
_safeTransfer	Internal	Can Modify State	-
_safeTransferFrom	Internal	Can Modify State	-

SigmaFeeDistributor			
Function Name	Visibility	Mutability	Modifiers
constructor	Public	Can Modify State	-

SigmaFeeDistributor			
_deposit	External	Can Modify State	-
_withdraw	External	Can Modify State	-
getPeriodReward	External	Can Modify State	nonReentrant
getReward	External	Can Modify State	nonReentrant
getRewardForOwner	External	Can Modify State	nonReentrant
incentivize	External	Can Modify State	nonReentrant
removeReward	External	Can Modify State	-
getRewardTokens	External	-	-
earned	External	-	-
getPeriod	Public	-	-
_getReward	Internal	Can Modify State	-
_getAllRewards	Internal	Can Modify State	-
_safeTransfer	Internal	Can Modify State	-
_safeTransferFrom	Internal	Can Modify State	-

SigmaVesting			
Function Name	Visibility	Mutability	Modifiers
constructor	Public	Can Modify State	-
addVestingSchedule	External	Can Modify State	onlyGovernance
removeVestingSchedule	External	Can Modify State	onlyGovernance
release	External	Can Modify State	nonReentrant
getReleasableAmount	Public	-	-
getVestingSchedule	External	-	-

SigmaVesting			
getBeneficiariesCount	External	-	-

VeFunderGauge			
Function Name	Visibility	Mutability	Modifiers
constructor	Public	Can Modify State	-
rewardsList	External	-	-
rewardsListLength	External	-	-
lastTimeRewardApplicable	Public	-	-
rewardData	External	-	-
getReward	Public	Can Modify State	nonReentrant
left	Public	-	-
whitelistReward	External	Can Modify State	-
removeRewardWhitelist	External	Can Modify State	-
notifyRewardAmount	External	Can Modify State	nonReentrant
isWhitelisted	Public	-	-
_safeTransfer	Internal	Can Modify State	-
_safeTransferFrom	Internal	Can Modify State	-

Minter			
Function Name	Visibility	Mutability	Modifiers
constructor	Public	Can Modify State	-
kickoff	External	Can Modify State	-
setGaugeActive	External	Can Modify State	onlyGovernance

Minter			
updatePeriod	External	Can Modify State	-
startEmissions	External	Can Modify State	-
updateEmissionsMultiplier	External	Can Modify State	onlyGovernance
calculateWeeklyEmissions	Public	View	-
releaseSigmaVesting	External	Can Modify State	-
getPeriod	Public	View	-
getEpoch	Public	View	-
_safeTransfer	Internal	Can Modify State	-

AccessHub			
Function Name	Visibility	Mutability	Modifiers
constructor	Public	Can Modify State	-
initialize	External	Can Modify State	initializer
reinit	External	Can Modify State	timelocked
initializeVoter	External	Can Modify State	timelocked
addVestingSchedule	External	Can Modify State	onlyRole
removeVestingSchedule	External	Can Modify State	onlyRole
setNewGovernorInVoter	External	Can Modify State	onlyRole
createSigmaGauge	External	Can Modify State	onlyRole
createVeFunderGauge	External	Can Modify State	onlyRole
governanceWhitelist	External	Can Modify State	onlyRole
killGauge	External	Can Modify State	onlyRole
reviveGauge	External	Can Modify State	onlyRole

AccessHub			
setEmissionsRatioInVoter	External	Can Modify State	onlyRole
retrieveStuckEmissionsToGovernance	External	Can Modify State	onlyRole
setSigmaGaugePreallocation	External	Can Modify State	onlyRole
createCLGaugeOveridden	External	Can Modify State	onlyRole
transferWhitelistInXShadow	External	Can Modify State	onlyRole
toggleXShadowGovernance	External	Can Modify State	onlyRole
operatorRedeemXShadow	External	Can Modify State	onlyRole
migrateOperator	External	Can Modify State	onlyRole
rescueTrappedTokens	External	Can Modify State	onlyRole
setExemptionToInXShadow	External	Can Modify State	onlyRole
setEmissionsMultiplierInMinter	External	Can Modify State	onlyRole
setGaugeActiveInMinter	External	Can Modify State	onlyRole
augmentGaugeRewardsForPair	External	Can Modify State	onlyRole
removeFeeDistributorRewards	External	Can Modify State	onlyRole
setCooldownExemption	External	Can Modify State	timelocked
setNewRebaseStreamingDuration	External	Can Modify State	timelocked
setNewVoteModuleCooldown	External	Can Modify State	timelocked
kickInactive	External	Can Modify State	onlyRole
execute	External	Can Modify State	timelocked
setNewTimelock	External	Can Modify State	timelocked

XShadow			
Function Name	Visibility	Mutability	Modifiers

XShadow			
constructor	Public	Can Modify State	-
pause	External	Can Modify State	onlyGovernance
unpause	External	Can Modify State	onlyGovernance
_update	Internal	Can Modify State	override
_isExempted	Internal	-	-
convertEmissionsToken	External	Can Modify State	whenNotPaused
rebase	External	Can Modify State	whenNotPaused
exit	External	Can Modify State	whenNotPaused
createVest	External	Can Modify State	whenNotPaused
exitVest	External	Can Modify State	whenNotPaused
operatorRedeem	External	Can Modify State	onlyGovernance
rescueTrappedTokens	External	Can Modify State	onlyGovernance
migrateOperator	External	Can Modify State	onlyGovernance
setExemption	External	Can Modify State	onlyGovernance
setExemptionTo	External	Can Modify State	onlyGovernance
getBalanceResiding	Public	-	-
usersTotalVests	Public	-	-
getVestInfo	Public	-	-
isExempt	External	-	-
shadow	External	-	-

Voter			
Function Name	Visibility	Mutability	Modifiers

Voter			
constructor	Public	Can Modify State	-
initialize	External	Can Modify State	initializer
setGlobalRatio	External	Can Modify State	onlyGovernance
reset	External	Can Modify State	-
_reset	Internal	Can Modify State	-
poke	External	Can Modify State	-
vote	External	Can Modify State	-
_vote	Internal	Can Modify State	-
_distribute	Internal	Can Modify State	-
getVotes	External	-	-
setGovernor	External	Can Modify State	onlyGovernance
whitelist	Public	Can Modify State	onlyGovernance
revokeWhitelist	Public	Can Modify State	onlyGovernance
killGauge	Public	Can Modify State	onlyGovernance
reviveGauge	Public	Can Modify State	onlyGovernance
stuckEmissionsRecovery	External	Can Modify State	onlyGovernance
whitelistGaugeRewards	External	Can Modify State	onlyGovernance
removeGaugeRewardWhitelist	External	Can Modify State	onlyGovernance
removeFeeDistributorReward	External	Can Modify State	onlyGovernance
getPeriod	Public	-	-
createSigmaGauge	External	Can Modify State	onlyGovernance
createVeFunderGauge	External	Can Modify State	onlyGovernance

Voter			
setSigmaGaugePreallocation	External	Can Modify State	onlyGovernance
claimIncentives	External	Can Modify State	-
claimRewards	External	Can Modify State	-
claimLegacyRewardsAndExit	External	Can Modify State	-
notifyRewardAmount	External	Can Modify State	-
distribute	Public	Can Modify State	nonReentrant
distributeForPeriod	Public	Can Modify State	nonReentrant
distributeAll	External	Can Modify State	-
batchDistributeByIndex	External	Can Modify State	-
getAllGauges	External	-	-
getAllFeeDistributors	External	-	-
isGauge	External	-	-
isFeeDistributor	External	-	-
_claimablePerPeriod	Internal	-	-
_getTotalPreallocationForPeriod	Internal	-	-
_getCurrentTotalSigmaGaugePreallocation	Internal	-	-

4.3 Vulnerability Summary

[N1] [Suggestion] Ignore the return value of the function

Category: Others

Content

1.In the Gauge contract, the `whitelistReward` and `removeRewardWhitelist` functions ignored the return values of `tokenWhitelists.add` and `tokenWhitelists.remove`.

- dao/contracts/SigmaGauge.sol#L239-L244, L247-L251

```
    function whitelistReward(address _reward) external {
        require(msg.sender == voter, NOT_AUTHORIZED());
        /// @dev voter checks for governance whitelist before allowing call
        tokenWhitelists.add(_reward);
        emit RewardWhitelisted(_reward, true);
    }

    function removeRewardWhitelist(address _reward) external {
        require(msg.sender == voter, NOT_AUTHORIZED());
        tokenWhitelists.remove(_reward);
        emit RewardWhitelisted(_reward, false);
    }
```

2.In the VeFunderGauge contract, the `constructor` and `whitelistReward` functions ignored the return values of `tokenWhitelists.add` and `tokenWhitelists.remove`.

- dao/contracts/VeFunderGauge.sol#L48-L61, L125-L130, L133-L137

```
    constructor(address _voter, address _receiver, uint256 _maxEmission) {
        //...
        tokenWhitelists.add(tempVoter.shadow());
        tokenWhitelists.add(tempVoter.xShadow());
    }

    function whitelistReward(address _reward) external {
        require(msg.sender == voter, NOT_AUTHORIZED());
        /// @dev voter checks for governance whitelist before allowing call
        tokenWhitelists.add(_reward);
        emit RewardWhitelisted(_reward, true);
    }

    function removeRewardWhitelist(address _reward) external {
        require(msg.sender == voter, NOT_AUTHORIZED());
        tokenWhitelists.remove(_reward);
        emit RewardWhitelisted(_reward, false);
    }
```

3.In the AccessHub contract, the initialize function ignores the return value of `_grantRole`.

- dao/contracts/AccessHub.sol#L62-L85

```
function initialize(InitParams calldata params) external initializer {
    //...
    /// @dev fee setter role given to treasury
    _grantRole(SWAP_FEE_SETTER, params.treasury);
    /// @dev operator role given to treasury
    _grantRole(PROTOCOL_OPERATOR, params.treasury);
    /// @dev initially give admin role to treasury
    _grantRole(DEFAULT_ADMIN_ROLE, params.treasury);
    /// @dev give timelock the admin role
    _grantRole(DEFAULT_ADMIN_ROLE, params.timelock);
}
```

4.In the Voter contract, the `killGauge`, `stuckEmissionsRecovery`, and `notifyRewardAmount` functions ignored the return values of `IERC20(shadow).transfer` and `IERC20(shadow).transferFrom`.

- dao/contracts/Voter.sol#L471-L507, L523-L544, L734-L745

```
function killGauge(address _gauge) public onlyGovernance {
    //...
    if (_claimable > 0) {
        /// @dev send to the governor contract
        IERC20(shadow).transfer(governor, _claimable);
    //...
}

function stuckEmissionsRecovery(
    address _gauge,
    uint256 _period
) external onlyGovernance {
    //...
        if (_claimable > 0) {
            IERC20(shadow).transfer(governor, _claimable);
    //...
}

function notifyRewardAmount(uint256 amount) external {
    //...
    IERC20(shadow).transferFrom(msg.sender, address(this), amount);
    //...
}
```

5.In the Voter contract, the `createSigmaGauge` and `createVeFunderGauge` functions ignored the return values of `IERC20(shadow).approve`, `IERC20(xShadow).approve`, `pools.add`, `gauges.add`, and

`feeDistributors.add`.

- dao/contracts/Voter.sol#L587-L630, L632-L671, L632-L671

```
    function createSigmaGauge(address _pool, uint256 _preallocationBps) external
onlyGovernance returns (address) {
        //...
        IERC20(shadow).approve(_gauge, type(uint256).max);
        IERC20(xShadow).approve(_gauge, type(uint256).max);
        //...
        pools.add(_pool);
        gauges.add(_gauge);
        feeDistributors.add(_feeDistributor);
        //...
    }

    function createVeFunderGauge(address _receiver, uint256 _maxEmission, address
_pool) external onlyGovernance returns (address) {
        //...
        IERC20(shadow).approve(_gauge, type(uint256).max);
        IERC20(xShadow).approve(_gauge, type(uint256).max);
        //...
        pools.add(_pool);
        gauges.add(_gauge);
        feeDistributors.add(_feeDistributor);
        //...
    }
```

Solution

It is recommended to check the return value of the function.

Status

Confirming

[N2] [Suggestion] Missing zero address check

Category: Others

Content

1.In the SigmaVesting contract, the `constructor` function lacks a zero address check for `_minter`.

- dao/contracts/SigmaVesting.sol#L40-L53

```
    constructor(
        address _sigma,
        address _xSigma,
        address _accessHub,
        address _minter
    ) {
        if (_sigma == address(0) || _xSigma == address(0) || _accessHub ==
address(0)) {
            revert ZeroAddress();
        }
        //...
        MINTER = _minter;
    }
```

2.In the Gauge contract, the `constructor` function lacks zero address checks for stake and voter.

- dao/contracts/SigmaGauge.sol#L46-L57

```
    constructor(address _stake, address _voter) {
        stake = _stake;
        voter = _voter;
        //...
    }
```

3.In the VeFunderGauge contract, the `constructor` function lacks zero address checks for `receiver` and `voter`.

- dao/contracts/VeFunderGauge.sol#L48-L61

```
    constructor(address _voter, address _receiver, uint256 _maxEmission) {
        voter = _voter;
        receiver = _receiver;
        //...
    }
```

4.In the SigmaVesting contract, the constructor function lacks a zero address check for `_minter`.

- dao/contracts/SigmaVesting.sol#L40-L53

```
    constructor(
        address _sigma,
```

```
        address _xSigma,
        address _accessHub,
        address _minter
    ) {
        if (_sigma == address(0) || _xSigma == address(0) || _accessHub ==
address(0)) {
            revert ZeroAddress();
        }
        //...
        MINTER = _minter;
    }
```

5.In the Minter contract, the `constructor` function lacks zero address checks for `_accessHub` and `_operator`.

- dao/contracts/Minter.sol#L53-L56

```
    constructor(address _accessHub, address _operator) {
        accessHub = _accessHub;
        operator = _operator;
    }
```

6.In the AccessHub contract, the `initialize` function lacks a zero address check for the address type parameter.

- dao/contracts/AccessHub.sol#L62-L85

```
    function initialize(InitParams calldata params) external initializer {
        /// @dev initialize all external interfaces
        timelock = params.timelock;
        treasury = params.treasury;
        voter = IVoter(params.voter);
        minter = IMinter(params.minter);
        xShadow = IXShadow(params.xShadow);
        voteModule = IVoteModule(params.voteModule);
        sigmaVesting = ISigmaVesting(params.sigmaVesting);

        /// @dev reference addresses
        sigmaGaugeFactory = params.sigmaGaugeFactory;
        veFunderGaugeFactory = params.veFunderGaugeFactory;
        sigmaFeeDistributorFactory = params.sigmaFeeDistributorFactory;
        //...
    }
```

7.In the AccessHub contract, the `reinit` function lacks a zero address check for the address type parameter.

- dao/contracts/AccessHub.sol#L87-L98

```
function reinit(InitParams calldata params) external timelocked {
    voter = IVoter(params.voter);
    minter = IMinter(params.minter);
    xShadow = IXShadow(params.xShadow);
    voteModule = IVoteModule(params.voteModule);
    sigmaVesting = ISigmaVesting(params.sigmaVesting);

    /// @dev reference addresses
    sigmaGaugeFactory = params.sigmaGaugeFactory;
    veFunderGaugeFactory = params.veFunderGaugeFactory;
    sigmaFeeDistributorFactory = params.sigmaFeeDistributorFactory;
}
```

8.In the Voter contract, the `constructor` and `initialize` functions lack zero address checks for address type parameters.

- dao/contracts/Voter.sol#L114-L116, L118-L151

```
constructor(address _accessHub) {
    accessHub = _accessHub;
}

function initialize(
    address _shadow,
    address _veFunderGaugeFactory,
    address _sigmaGaugeFactory,
    address _sigmaFeeDistributorFactory,
    address _minter,
    address _msig,
    address _xShadow,
    address _voteModule,
    uint256 _maxTotalSigmaGaugePreallocation,
    uint256 _maxVeFunderGaugeCap
) external initializer {
    //...
    shadow = _shadow;
    veFunderGaugeFactory = _veFunderGaugeFactory;
    sigmaGaugeFactory = _sigmaGaugeFactory;
    sigmaFeeDistributorFactory = _sigmaFeeDistributorFactory;
    minter = _minter;
    xShadow = _xShadow;
    governor = _msig;
    voteModule = _voteModule;
```

```
        maxVeFunderGaugeCap = _maxVeFunderGaugeCap;
        //...
    }
```

Solution

It is recommended to add zero address check.

Status

Confirming

[N3] [Medium] The maximum emission limit of the reward has expired

Category: Design Logic Audit

Content

In the VeFunderGauge contract, the `getReward` function did not accumulate and update the `emission` state variable after distributing rewards, causing `remainReward` to always be the value of `maxEmission` on each call. This completely invalidated the `maxEmission` limit on the cumulative emission of `shadow` and `xShadow` tokens.

- dao/contracts/VeFunderGauge.sol#L89-L111

```
    function getReward(
        address account,
        address[] calldata tokens
    ) public nonReentrant {
        /// @dev ensure calls from the account or the voter address
        require(msg.sender == account || msg.sender == voter, NOT_AUTHORIZED());
        /// @dev loop through the tokens
        uint256 remainReward = maxEmission - emission;
        for (uint256 i; i < tokens.length; i++) {
            /// @dev fetch the stored rewards for the user for current index's token
            uint256 _reward = IERC20(tokens[i]).balanceOf(address(this));
            if (tokens[i] == xShadow || tokens[i] == shadow) {
                _reward = Math.min(_reward, remainReward);
                remainReward -= _reward;
            /// @dev if the stored rewards are greater than zero
            if (_reward > 0) {
                /// @dev transfer the expected rewards
                _safeTransfer(tokens[i], receiver, _reward);
                emit ClaimRewards(receiver, tokens[i], _reward);
            }
        }
    }
```

Solution

It is recommended to update the emission value uniformly after the loop ends.

Status

Confirming

[N4] [Medium] Risk of excessive token minting

Category: Design Logic Audit

Content

In the SigmaVesting contract, the `addVestingSchedule` function only verifies that `totalAmount > 0` when creating a vesting schedule, but fails to check whether `totalAmount` plus the current token supply exceeds `MAX_SUPPLY` defined in the Minter contract. Because the `releaseSigmaVesting` function directly calls `shadow.mint()` to mint tokens without checking supply, when the total vesting schedule exceeds the maximum supply, the actual minting amount exceeds the intended limit, undermining the supply design of the token economics model.

- dao/contracts/SigmaVesting.sol#L57-L90

```
    function addVestingSchedule(
        address _beneficiary,
        address _tokenAddress,
        uint8 _category,
        UnlockEntry[] calldata _entries
    ) external onlyGovernance {
        if (_beneficiary == address(0)) revert ZeroAddress();
        if (_tokenAddress != sigma && _tokenAddress != xSigma) revert
InvalidTokenAddress();
        if (_category < 1 || _category > 6) revert InvalidCategory();
        if (vestingSchedules[_beneficiary][_tokenAddress].isInitialized) revert
ScheduleAlreadyExists();

        uint256 totalAmount = 0;
        uint64 lastTimestamp = 0;
        for (uint256 i = 0; i < _entries.length; i++) {
            if (_entries[i].timestamp <= lastTimestamp) revert UnorderedTimestamps();
            totalAmount += _entries[i].amount;
            lastTimestamp = _entries[i].timestamp;
        }
        if (totalAmount == 0) revert ZeroTotalAmount();
```

```
        // Add beneficiary to iterable list if they are new
        if (!isBeneficiary[_beneficiary]) {
            isBeneficiary[_beneficiary] = true;
            beneficiaries.push(_beneficiary);
        }

        VestingSchedule storage schedule = vestingSchedules[_beneficiary]
[_tokenAddress];
        schedule.isInitialized = true;
        schedule.vestingCategory = _category;
        schedule.totalAmount = totalAmount;
        schedule.unlockEntries = _entries;

        emit ScheduleAdded(_beneficiary, _tokenAddress, totalAmount, _category);
    }
```

Solution

We recommend verifying that shadow.totalSupply + totalAmount <= MAX_SUPPLY in the addVestingSchedule function. Modify the release function logic to transfer tokens from the contract balance instead of dynamic minting, automatically releasing locked tokens based on the unlocking time. Also, add token burning logic to the removeVestingSchedule function.

Status

Confirming

[N5] [Suggestion] Missing event records

Category: Others

Content

In the AccessHub contract, the `setGaugeActiveInMinter` function lacks event logging when setting key variables.

- dao/contracts/AccessHub.sol#L305-L309

```
    function setGaugeActiveInMinter(
        bool _isGaugeActive
    ) external onlyRole(PROTOCOL_OPERATOR) {
        minter.setGaugeActive(_isGaugeActive);
    }
```

Solution

It is recommended to add time records.

Status

Confirming

[N6] [Medium] Risk of excessive authority

Category: Authority Control Vulnerability Audit

Content

1.In the Minter contract, the `operator` role can call the `kickoff` function to set the key contract address and initial issuance parameters at one time, and can call the `startEmissions` function to start the entire token issuance process.

- dao/contracts/Minter.sol#L59-L88, L133-L144

```
    function kickoff(
        address _shadow,
        address _voter,
        uint256 _initialWeeklyEmissions,
        uint256 _initialMultiplier,
        uint256 _multiplierUpdatePeriod,
        address _xShadow,
        address _sigmaVesting
    ) external {}

    function startEmissions() external {}
```

2.In the AccessHub contract, the `DEFAULT_ADMIN_ROLE` role can grant and revoke all other roles and has the `kickInactive` function permission to kick out inactive users who have not voted and reset their voting status.

- dao/contracts/AccessHub.sol#L380-L402

```
    function kickInactive(
        address[] calldata _nonparticipants
    ) external onlyRole(DEFAULT_ADMIN_ROLE) {}
```

3.In the AccessHub contract, the `PROTOCOL_OPERATOR` role has operational management permissions, including: managing the addition and removal of SigmaVesting, setting the governor of the Voter contract, creating and

managing SigmaGauge and VeFunderGauge, whitelist management (token whitelist, reward whitelist), pause/unpause Gauge, setting emission ratios and pre-allocation, controlling xShadow's transfer whitelist and pause/unpause functions, operator redemption and migration, adjusting Minter's emission multiples, and switching Gauge token emission status, among other core functions.

- dao/contracts/AccessHub.sol#L130-L137, L140-L142, L147-L152, L154-L159, L161-L167, L170-L187, L190-L200, L203-L211, L214-L218, L221-L226, L229-L234, L236-L242, L247-L254, L257-L262, L265-L269, L272-L276, L279-L284, L287-L294, L299-L303, L305-L309, L314-L338, L340-L351

```
    function addVestingSchedule(
        address _beneficiary,
        address _tokenAddress,
        uint8 _category,
        ISigmaVesting.UnlockEntry[] calldata _entries
    ) external onlyRole(PROTOCOL_OPERATOR) {}

    function removeVestingSchedule(address _beneficiary, address _tokenAddress)
external onlyRole(PROTOCOL_OPERATOR) {}

    function setNewGovernorInVoter(address _newGovernor) external
onlyRole(PROTOCOL_OPERATOR) {}

    function createSigmaGauge(address _pool, uint256 _preallocationBps) external
onlyRole(PROTOCOL_OPERATOR) {}

    function createVeFunderGauge(address _receiver, uint256 _maxEmission, address
_pool) external onlyRole(PROTOCOL_OPERATOR) {}

    function governanceWhitelist(address[] calldata _token, bool[] calldata
_whitelisted) external onlyRole(PROTOCOL_OPERATOR) {}

    function killGauge(address[] calldata _pairs) external
onlyRole(PROTOCOL_OPERATOR) {}

    function reviveGauge(address[] calldata _pairs) external
onlyRole(PROTOCOL_OPERATOR) {}

    function setEmissionsRatioInVoter(uint256 _pct) external
onlyRole(PROTOCOL_OPERATOR) {}

    function retrieveStuckEmissionsToGovernance(address _gauge, uint256 _period)
external onlyRole(PROTOCOL_OPERATOR) {}

    function setSigmaGaugePreallocation(address _gauge, uint256 _preallocationBps)
```

```
external onlyRole(PROTOCOL_OPERATOR) {}

    function createCLGaugeOveridden(address tokenA, address tokenB, int24
tickSpacing) external onlyRole(PROTOCOL_OPERATOR) {}

    function transferWhitelistInXShadow(address[] calldata _who, bool[] calldata
_whitelisted) external onlyRole(PROTOCOL_OPERATOR) {}

    function toggleXShadowGovernance(bool enable) external
onlyRole(PROTOCOL_OPERATOR) {}

    function operatorRedeemXShadow(uint256 _amount) external
onlyRole(PROTOCOL_OPERATOR) {}

    function migrateOperator(address _operator) external onlyRole(PROTOCOL_OPERATOR)
{}

    function rescueTrappedTokens(address[] calldata _tokens, uint256[] calldata
_amounts) external onlyRole(PROTOCOL_OPERATOR) {}

    function setExemptionToInXShadow(address[] calldata _who, bool[] calldata
_whitelisted) external onlyRole(PROTOCOL_OPERATOR) {}

    function setEmissionsMultiplierInMinter(uint256 _multiplier) external
onlyRole(PROTOCOL_OPERATOR) {}

    function setGaugeActiveInMinter(bool _isGaugeActive) external
onlyRole(PROTOCOL_OPERATOR) {}

    function augmentGaugeRewardsForPair(address[] calldata _pools, address[] calldata
_rewards, bool[] calldata _addReward) external onlyRole(PROTOCOL_OPERATOR) {}

    function removeFeeDistributorRewards(address[] calldata _pools, address[]
calldata _rewards) external onlyRole(PROTOCOL_OPERATOR) {}
```

Solution

In the short term, transferring owner ownership to multisig contracts is an effective solution to avoid single-point risk. But in the long run, it is a more reasonable solution to implement a privilege separation strategy and set up multiple privileged roles to manage each privileged function separately. And the authority involving user funds should be managed by the community, and the EOA address can manage the authority involving emergency contract suspension. This ensures both a quick response to threats and the safety of user funds.

Status

Confirming

[N7] [Suggestion] Improper variable declaration

Category: Others

Content

1.In the Voter contract, `accessHub` should be immutable.

- dao/contracts/Voter.sol#L44

```
    address public accessHub;
```

2.In the Minter contract, `operator` and `accessHub` should be immutable.

- dao/contracts/Minter.sol#L36, L38

```
    address public operator;
    address public accessHub;
```

Solution

It is recommended to add appropriate modifiers to variables.

Status

Confirming

[N8] [Medium] Risk of excess preallocation amount

Category: Design Logic Audit

Content

In the Voter contract, when the `killGauge` function kills the Gauge, the `preallocationBps` in its `sigmaGaugePreallocation` mapping is not cleared. When the `reviveGauge` function revives the Gauge, it does not check whether the recovered `preallocationBps` exceeds the total limit. This may cause the actual effective preallocation amount to exceed the limit of `maxTotalSigmaGaugePreallocation`.

- dao/contracts/Voter.sol#L471-L507, L509-L520

```
    function killGauge(address _gauge) public onlyGovernance {
        /// @dev ensure the gauge is alive already, and exists
        require(
            isAlive[_gauge] && gauges.contains(_gauge),
```

```
            GAUGE_INACTIVE(_gauge)
        );
        /// @dev set the gauge to dead
        isAlive[_gauge] = false;
        address pool = poolForGauge[_gauge];
        /// @dev fetch the last distribution
        uint256 _lastDistro = lastDistro[_gauge];
        /// @dev fetch the current period
        uint256 currentPeriod = getPeriod();
        /// @dev placeholder
        uint256 _claimable;
        /// @dev loop through the last distribution period up to and including the
current period
        for (uint256 period = _lastDistro; period <= currentPeriod; ++period) {
            /// @dev if the gauge isn't distributed for the period
            if (!gaugePeriodDistributed[_gauge][period]) {
                uint256 additionalClaimable = _claimablePerPeriod(pool, period);
                _claimable += additionalClaimable;

                /// @dev prevent gaugePeriodDistributed being marked true when the
minter hasn't updated yet
                if (additionalClaimable > 0) {
                    gaugePeriodDistributed[_gauge][period] = true;
                }
            }
        }
        /// @dev if there is anything claimable left
        if (_claimable > 0) {
            /// @dev send to the governor contract
            IERC20(shadow).transfer(governor, _claimable);
        }
        /// @dev update last distribution to the current period
        lastDistro[_gauge] = currentPeriod;
        emit GaugeKilled(_gauge);
    }

    function reviveGauge(address _gauge) public onlyGovernance {
        /// @dev ensure the gauge is dead and exists
        require(
            !isAlive[_gauge] && gauges.contains(_gauge),
            ACTIVE_GAUGE(_gauge)
        );
        /// @dev set the gauge to alive
        isAlive[_gauge] = true;
        /// @dev update last distribution to the current period
        lastDistro[_gauge] = getPeriod();
        emit GaugeRevived(_gauge);
    }
```

Solution

It is recommended to clear the preallocation amount of SigmaGauge in the killGauge function and reallocate the preallocation amount of SigmaGauge in the reviveGauge function.

Status

Confirming

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002509300001	SlowMist Security Team	2025.09.28 - 2025.09.30	Medium Risk

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 4 medium risk, 4 suggestion vulnerabilities. All the findings were fixed. The code was not deployed to the mainnet.

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