SALUS SECURITY

OCT 2023

CODE SECURITY ASSESSMENT

INK FINANCE

Overview

Project Summary

- Name: Ink Finance - Incremental Audit
- Platform: EVM-compatible Chains
- Language: Solidity
- Repository:
    - https://github.com/Ink-Finance-Inc/v2-governance-core
    - https://github.com/Ink-Finance-Inc/v3-economy-core
- Audit Scope: See Appendix - 1

Project Dashboard

Application Summary

Name	Ink Finance - Incremental Audit
Version	v1
Type	Solidity
Dates	Oct 26 2023
Logs	Oct 26 2023

Vulnerability Summary

Total High-Severity issues	11
Total Medium-Severity issues	2
Total Low-Severity issues	2
Total informational issues	9
Total	24

Contact

E-mail: support@salusec.io

Risk Level Description

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for clients' reputations or serious financial implications for clients and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental to the client's reputation if exploited, or is reasonably likely to lead to a moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or defense in depth.

Content

Introduction

1.1 About SALUS

At Salus Security, we are in the business of trust.

We are dedicated to tackling the toughest security challenges facing the industry today. By building foundational trust in technology and infrastructure through security, we help clients to lead their respective industries and unlock their full Web3 potential.

Our team of security experts employ industry-leading proof-of-concept (PoC) methodology for demonstrating smart contract vulnerabilities, coupled with advanced red teaming capabilities and a stereoscopic vulnerability detection service, to deliver comprehensive security assessments that allow clients to stay ahead of the curve.

In addition to smart contract audits and red teaming, our Rapid Detection Service for smart contracts aims to make security accessible to all. This high calibre, yet cost-efficient, security tool has been designed to support a wide range of business needs including investment due diligence, security and code quality assessments, and code optimisation.

We are reachable on Telegram (https://t.me/salusec), Twitter (https://twitter.com/salus_sec), or Email (support@salusec.io).

1.2 Audit Breakdown

The objective was to evaluate the repository for security-related issues, code quality, and adherence to specifications and best practices. Possible issues we looked for included (but are not limited to):

- Risky external calls
- Integer overflow/underflow
- Transaction-ordering dependence
- Timestamp dependence
- Access control
- Call stack limits and mishandled exceptions
- Number rounding errors
- Centralization of power
- Logical oversights and denial of service
- Business logic specification
- Code clones, functionality duplication

1.3 Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release and does not give any warranties on finding all possible security issues with the given smart contract(s) or blockchain software, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues.

Findings

2.1 Summary of Findings

ID	Title	Severity	Category	Status
1	Anyone can upgrade EconomyEngineV1Factory's main StakingEngine	High	Access Control	Pending
2	Any voter can block vote process	High	Denial of Service	Pending
3	Missing access control for liquidate()	High	Access Control	Pending
4	Anyone can transfer any asset by calling distribute()	High	Access Control	Pending
5	Anyone can claim investments for any user	High	Access Control	Pending
6	Attacker could DoS tallyUpFund() by directly sending tokens to InkFund	High	Denial of Service	Pending
7	Anyone can create escrow trade	High	Access Control	Pending
8	Anyone can approve any trade	High	Access Control	Pending
9	Anyone can set PayrollUCV controller	High	Access Control	Pending
10	Missing access control for unpledge() and unpledgeAll()	High	Access Control	Pending
11	Anyone can make earlyRedeem for others	High	Access Control	Pending
12	One board member could be added multiple times	Medium	Data Validation	Pending
13	tallyUpFund() can be DoSed by user creating active trades	Medium	Data Validation	Pending
14	Multiple freezing of NFTs possible	Low	Denial of Service	Pending
15	Incomplete duty removal in remmoveDuty()	Low	Business Logic	Pending
16	Any agent can upgrade the implementation of an InkBeaconProxy	Informational	Access Control	Pending
17	New staking basket cannot be created after defaultStakingBasket is set	Informational	Business Logic	Pending
18	Anyone can call claim rewards for other user's item	Informational	Access Control	Pending
19	Weak ID generation process	Informational	Business Logic	Pending
20	Could use safeTransfer()/safeTransferFrom() instead of transfer()/transferFrom()	Informational	Data Validation	Pending

21	Unused state variables	Informational	Redundancy	Pending
22	Typo	Informational	Typo	Pending
23	Redundant Code	Informational	Redundancy	Pending
24	Inconsistent function name and logic in canVoteChangeResult()	Informational	Business Logic	Pending

2.2 Notable Findings

Significant flaws that impact system confidentiality, integrity, or availability are listed below.

1. Anyone can upgrade EconomyEngineV1Factory's main StakingEngine

Severity: High	Category: Access Control
Target: - v3-economy-core/contracts/EconomyEngineV1Factory.sol	

Description

In EconomyEngineV1Factory contract, the _stakingEngine state variable stores the proxy address of the Ink Main Staking Engine, and the upgradeStakingEngine() function is used to upgrade the _stakingEngine to a new implementation.

v3-economy-core/contracts/EconomyEngineV1Factory.sol:L244-L250

```
function upgradeStakingEngine(address newImplementation) public {
    if (_stakingEngine == address(0)) {
        revert INK_ERROR(1998); /// Invalid staking engine proxy
    }
    proxyAdmin.upgradeTo(_stakingEngine, newImplementation);
    emit StakingEngineUpgraded(msg.sender, newImplementation, block.timestamp);
}
```

However, the upgradeStakingEngine() function lacks access control, meaning malicious users can exploit it to upgrade _stakingEngine to a malicious implementation. This can lead to the compromise of the contract and the loss of user funds.

Recommendation

Consider adding proper access control to the upgradeStakingEngine() function.

2. Any voter can block vote process

Severity: High	Category: Denial of Service
Target: - v2-governance-core\contracts\committee\TheBoard.sol	

Description

When a proposer makes a proposal, the status of proposal is set to “PENDING” and everyone with voter duty can vote for this.

However, voters can call the tallyVotes() function while the voting period is ongoing. If there are insufficient "pass" votes, calling tallyVotes() will change the proposal status to "DENY," preventing further voting.

v2-governance-core\contracts\committee\TheBoard.sol:L227-L232

```
function _tallyVotes(VoteIdentity memory identity, bytes memory data)
    internal
{
    ...
    VoteInfo storage voteInfo = _voteInfos[identity._getIdentityID()];
    if (passOrNot) {
        voteInfo.status = VoteStatus.AGREE;
    } else {
        voteInfo.status = VoteStatus.DENY;
    }
    ...
}
```

Recommendation

It is recommended to implement vote time duration control within the tallyVotes() function. It should ensure that proposals are not prematurely moved to the DENY status before the voting period is complete.

3. Missing access control for liquidate()

Severity: High	Category: Access Control

Target:
- v2-governance-core\contracts\products\InkFund.sol
- v2-governance-core\contracts\products\funds\FundManager.sol

Description

v2-governance-core\contracts\products\funds\FundManager.sol:L1256-L1267

```
function liquidateFund(bytes32 fundID) external override {
    //...
    if (!_isCommitteeOperator(3, msg.sender)) {
        revert INK_ERROR(3001);
    }

    IFund(_funds[fundID]).liquidate();
}
```

When liquidating funds, it's expected for “fundLiquidator” to call the liquidateFund() function in the FundManager contract. This function, in turn, calls the liquidate() function in the InkFund contract.

v2-governance-core\contracts\products\InkFund.sol:L970-L972

```
function liquidate() external override {
    _isLiqudating = true;
}
```

However, the liquidate() function in the InkFund contract is an external function without proper access control, allowing anyone to set the _isLiqudating state to true.

Recommendation

It is recommended to add access control within the liquidate() in InkFund.sol. For example, the access should be restricted to only the FundManager contract.

4. Anyone can transfer any asset by calling distribute()

Severity: High	Category: Access Control

Target:
- v2-governance-core\contracts\products\InkFund.sol

Description

In the FundManager.sol, the claimPrincipalAndProfit() calls the distribute() in the InkFund.sol only under certain conditions. However, the distribute() function does not have any access control checks. This means that anyone can call the distribute function and specify arbitrary values for the token address, recipient, and amount.

v2-governance-core\contracts\products\InkFund.sol:L974-L983

```
function distribute(
    address owner,
    address token,
    uint256 amount
) external override {
    // _frozened = _frozened - amount;
    _unfrozenAsset(token, amount, 20, 0);

    _transferTo(owner, token, 20, 0, amount, "");
}
```

Recommendation

It is recommended to implement proper access control checks in the distribute() function.

5. Anyone can claim investments for any user

Severity: High	Category: Access Control

Target:
- v2-governance-core\contracts\products\InkFund.sol

Description

In the InkFund.sol contract, the claimInvestment() function allows for the redemption of funds for a specified investor. It is intended to be called by FindManager.sol during withdrawPrincipal() or claimPrincipalAndProfit(). However, it is missing an access control modifier, which means that anyone can call this function, pass custom params and steal assets.

v2-governance-core\contracts\products\InkFund.sol:L597

```
function claimInvestment(address investor) external override {
    ...
}
```

Recommendation

It is recommended to add an access modifier to the claimInvestment() function.

6. Attacker could DoS tallyUpFund() by directly sending tokens to InkFund

Severity: High	Category: Denial of Service

Target:
- v2-governance-core\contracts\products\funds\FundManager.sol
- v2-governance-core\contracts\products\InkFund

Description

In the FundManager.sol, the tallyUpFund() function checks whether a fund has 0 balance and 0 escrow trades before dissolving the fund. However, there is a potential denial-of-service (DoS) risk due to the execution of hasLeftBalance() within tallyUpFund(). An attacker can send whitelisted tokens/native to InkFund, which triggers error 3016 for tallyUpFund(). This subsequently DoS calls to the dissolve() function.

v2-governance-core\contracts\products\funds\FundManager.sol:L1198-L1206

```
function tallyUpFund(bytes32 fundID) external override {
    ...
    if (
        IFund(_funds[fundID]).hasLeftBalance(
            _tokenWhitelist[fundID].at(i),
            token.tokenType,
            token.tokenIdentity
        )
    ) {
        revert INK_ERROR(3016);
    }
    ...
}
```

v2-governance-core\contracts\products\InkFund.sol:L210-L213

```
function hasLeftBalance(address tokenAddress, uint256 tokenType, int256 tokenIdentity)
external view override returns (bool) {
    ...
    if (tokenType == 20) {
        if (tokenAddress == address(0)) {
            return address(this).balance > 0;
        }
        ...
    }
}
```

Recommendation

It is recommended to modify the checking logic before tallying up funds. For example, use storage variables to track the accounting of tokens and check these internal accountings.

7. Anyone can create escrow trade

Severity: High	Category: Access Control

Target:
- v2-governance-core\contracts\products\EscrowManager.sol

Description

The createEscrowTrade() function in the EscrowManager.sol contract is designed to create escrow trades. However, it lacks the necessary access control modifiers. As a result, an attacker can create escrow trades with unfair parameters (amount/price) for a fund.

v2-governance-core\contracts\products\EscrowManager.sol:L86-L93

```
function createEscrowTrade(
    bytes32 fundID,
    address initiator,
    address seller,
    IFundInfo.Asset[] memory payForAsset,
    IFundInfo.Asset[] memory buyAsset,
    uint256 duration
) external override returns (bytes32 tradeID) {
    ...
}
```

Recommendation

The createEscrowTrade() function should only be called from the createTrade() function within the FundManager.sol. It is recommended to add an appropriate modifier to prevent unauthorized access.

8. Anyone can approve any trade

Severity: High	Category: Access Control

Target:
- v2-governance-core\contracts\products\EscrowManager.sol

Description

The approveTrade() function in EscrowManager.sol is responsible for approving or disapproving trades. However, it misses an access modifier (should be only fundManager with RiskManager rights) and enables unauthorized entities to influence the approval status of trades. This could lead to various adverse outcomes. For example, bad trades could be approved, affecting the fund's performance, or every trade could be disapproved, causing a DoS attack.

v2-governance-core\contracts\products\EscrowManager.sol:L117-L128

```
function approveTrade(bytes32 tradeID, EscrowTradeStatus status)
    external
    override
{
    if (status == EscrowTradeStatus.Approved) {
        _trades[tradeID].state = EscrowTradeStatus.OutboundCommitted;
        _trades[tradeID].startTime = block.timestamp;
    } else _trades[tradeID].state = EscrowTradeStatus.Disapproved;
}
```

Recommendation

It is recommended to add an access modifier to approveTrade() so only the person with risk management rights could call it.

9. Anyone can set PayrollUCV controller

Severity: High	Category: Access Control
Target: - v2-governance-core\contracts\ucv\PayrollUCV.sol	

Description

The PayrollUCV.sol exhibits a vulnerability due to a missing access control modifier in the setUCVController() function. This issue can potentially lead to unauthorized access to admin functions like transferTo(), so an attacker can steal all assets.

v2-governance-core\contracts\ucv\PayrollUCV.sol:L61-L63

```
function setUCVController(address controller_) external override {
    _ucvController = controller_;
}
```

Recommendation

It is recommended to add an access modifier to setUCVController().

10. Missing access control for unpledge() and unpledgeAll()

Severity: High	Category: Access Control

Target:
- v3-economy-core\contracts\engines\pledge\PledgeEngine.sol
- v3-economy-core\contracts\mockup\DaoGovernance.sol

Description

The unpledge() and unpledgeAll() functions in PledgeEngine.sol should be called by DAO from DaoGovernance.sol to release pledges from voting. However, it lacks proper access control. As a result, everyone can call these functions and unpledge assets from any voter.

v3-economy-core\contracts\engines\pledge\PledgeEngine.sol:L165-L178

```
function unpledge(
    address daoContract,
    address daoUser,
    bytes32 proposalId
) external override returns (uint256) {
    if (daoContract == address(0) || daoUser == address(0)) {
        revert INK_ERROR(2042); /// PE: Not allowed to be address 0
    }
    if (IDaoGovernance(daoContract).getProposalStatus(proposalId) != 1) {
revert INK_ERROR(2049); /// PE: DAO didn't allow to unpledge
    }

    return _unpledge(daoContract, daoUser, proposalId);
}
```

v3-economy-core\contracts\engines\pledge\PledgeEngine.sol:L165-L178

```
function unpledgeAll(address daoContract, address daoUser) external override returns
(uint256) {
    if (daoContract == address(0) || daoUser == address(0)) {
        revert INK_ERROR(2042); /// PE: Not allowed to be address 0
    }

    uint256 unpledgedValue = 0;
    /// 1. in the case of voter
    for (uint256 i = 0; i < _userProposalIds[daoContract][daoUser].length; i++) {
        LPledgeEngine.PledgeItemInfo memory pledgeItem = _userProposalPledges[
            _userProposalIds[daoContract][daoUser][i]
        ];
        if (IDaoGovernance(daoContract).getProposalStatus(pledgeItem.proposalId) == 1) {
            unpledgedValue += _unpledge(daoContract, daoUser, pledgeItem.proposalId);
        }
    }

    /// 2. in the case of manager
    if (!IDaoGovernance(daoContract).hasAnyDuty(daoUser)) {
        uint256 unpledged = _unpledge(daoContract, daoUser, UNDEFINED_DUTYID);
        unpledgedValue += unpledged;
    }

    return unpledgedValue;
}
```

Recommendation

It is recommended to add access modifiers to the unpledge() and unpledgeAll() functions.

11. Anyone can make earlyRedeem for others

Severity: High	Category: Access Control

Target:
- v3-economy-core\contracts\engines\staking\StakingEngine.sol

Description

The earlyRedeem() function allows unstaking an item belonging to a staker. However, unlike the unstake() function, the earlyRedeem() function does not verify if the msg.sender is the rightful staker linked to the specified itemId. As a result, malicious users can trigger early redemption for other users' items.

v3-economy-core\contracts\engines\staking\StakingEngine.sol:L498-L502

```
function earlyRedeem(
    address basket,
    bytes32 itemId,
    uint256 amount
) external override validBasket(basket) {
    ...
}
```

Recommendation

It is recommended to verify if msg.sender is the rightful staker in earlyRedeem().

12. One board member could be added multiple times

Severity: Medium	Category: Data Validation

Target:
- v2-governance-core\contracts\bases\BaseDAO.sol

Description

During the DAO creation process, the creator must fill VoteSettingInfo especially _agreeSeatsOfTheBoard.

Then the creator should send an array of members with their duties, like [0x01, VOTER]. So every mentioned user will have appropriate rights.

However, the creator could potentially input the same user multiple times and bypass the minimum seat limit by doing so. For instance, if the minimum seats required for a board are 5, the creator could input an address 5 times as board members.

Recommendation

Consider adding a validation process if a member was added before.

13. tallyUpFund() can be DoSed by user creating active trades

Severity: Medium	Category: Denial of Service

Target:
- v2-governance-core\contracts\products\funds\FundManager.sol

Description

v2-governance-core\contracts\products\funds\FundManager.sol:L1176-L1188

```
function tallyUpFund(bytes32 fundID) external override {
    ...

    // disapproved, released, completed
    address escrowManager = _fundEscrow[fundID];

    // all trades should be finished
    if (escrowManager != address(0)) {
        // require(
        //     !IEscrowManager(escrowManager).hasUnfinishedTrade(fundID),
        //     "This fund has pending trade(s)"
        // );

        if (IEscrowManager(escrowManager).hasUnfinishedTrade(fundID)) {
            revert INK_ERROR(3015);
        }
    }

    ...
}
```

The tallyUpFund() function in the FundManager contract checks if the escrowManager has any unfinished trades. If there are any, the transaction will fail.

As a result, a person who has rights to call EscrowManager.sol's createEscrowTrade() function can DoS the tallyUpFund() function and make dissolve() uncallable so no one can share profit.

Recommendation

Consider adding rights to fundAdmin to close manually escrow trades(unfrozen and send back to buyers/sellers) before tallyUpFund().

14. Multiple freezing of NFTs possible

Severity: Low	Category: Data Validation

Target:
- v2-governance-core\contracts\products\InkFund.sol

Description

In the _frozenAsset function, when the token type is EIP721, tokens can be frozen if the owner of the NFT is the contract itself.

However, the logic does not account for scenarios where an NFT is already frozen, which means that calling _frozenAsset multiple times for the same NFT results in an incorrect count of frozen assets. The same issue appears to occur in the _unfrozenAsset() function.

v2-governance-core\contracts\products\InkFund.sol:L159-L171

```
function _frozenAsset(
    address token,
    uint256 amount,
    uint256 tokenType,
    uint256 tokenIdentity
    ) internal {
        if (tokenType == 20) {
            ...
        } else if (tokenType == 721) {
            // require(
            //         IERC721(token).ownerOf(tokenIdentity) == address(this),
            //         "not NFT owner"
            // );
            if (IERC721(token).ownerOf(tokenIdentity) != address(this)) {
                // revert("not NFT owner");
                revert INK_ERROR(3030);
            }

            _frozenedAssets[token].frozenedToken[tokenIdentity] = 1;
            _frozenedAssets[token].frozened += 1;
        }
        ...
}
```

Recommendation

It is recommended to implement checks to ensure that an NFT is not frozen multiple times, which would lead to a more accurate count of frozen assets.

15. Incomplete duty removal in remmoveDuty()

Severity: Low	Category: Business Logic

Target:
- v2-governance-core\contracts\bases\BaseDAO.sol

Description

The remmoveDuty function is responsible for removing a duty associated with an account. However, it lacks a check to verify whether the address no longer holds any duties after the removal.

If this is the case, the account should be removed from _daoMembersWithDuties. But the code does not include any operation to remove elements from _daoMembersWithDuties.

v2-governance-core\contracts\bases\BaseDAO.sol:L847-L857

```
function remmoveDuty(address account, bytes32 dutyID)
    external
    override
    onlyAgent
{
    EnumerableSet.AddressSet storage memberOwnedDuty = _dutyMembers[dutyID];
    if (memberOwnedDuty.contains(account)) {
        memberOwnedDuty.remove(account);
        _dutyCounts[account] -= 1;
    }
}
```

Recommendation

Consider adding a check in the remmoveDuty() function. If an account no longer holds any duties after removing a duty, remove the account from the _daoMembersWithDuties data structure.

2.3 Informational Findings

16. Any agent can upgrade the implementation of an InkBeaconProxy

Severity: Informational	Category: Access Control
Target: - v2-governance-core\contracts\bases\BaseDAO.sol	

Description

The updateContract() function in BaseDAO.sol is intended to change the logic address corresponding to a proxy contract. It is restricted by the onlyAgent modifier, meaning that only addresses in the agents list can call it.

v2-governance-core\contracts\bases\BaseDAO.sol:L345-L355

```
function updateContract(address proxyAddress, address newImplement)
    external
    override
    onlyAgent
{
    (, address newBeaconAddress) = IFactoryManager(_factoryAddress)
        .getBeaconKeyAndAddress(proxyAddress, newImplement);
    InkBeaconProxy(payable(proxyAddress)).upgradeTo(
    address(newBeaconAddress)
    );
}
```

However, we think that this access control is not strict enough. Any agent contract can call this function to change the logic for other agent contracts. If, let's say, a flawed agent contract that is controlled by a malicious user is introduced to the system during further development, this malicious agent could potentially upgrade the implementation of other agents, thereby compromising the entire system.

Recommendation

It is recommended to enhance the access control mechanisms for the updateContract() function. Access should be restricted to only the agent contract that is associated with the proxy contract being updated. One way to achieve this is to ensure that proxyAddress is derived internally within the function, rather than being a parameter.

17. New staking basket cannot be created after defaultStakingBasket is set

Severity: Informational	Category: Business Logic
Target: - v3-economy-core\contracts\engines\staking\StakingEngine.sol	

Description

In StakingEngine.sol, the createStakingBasket() function allows the owner of the engine to create new staking baskets.

v3-economy-core\contracts\engines\staking\StakingEngine.sol:L270-L272

```
function createStakingBasket(uint256 termDays, uint256 weight) external override
onlyOwner {
    ...
    if (defaultStakingBasket != address(0)) {
        revert INK_ERROR(2009); /// SE: Already created staking basket with same
information
    }
    ...
}
```

However, the above check prevents the creation of new staking baskets if the defaultStakingBasket value is non-zero. This means that once the defaultStakingBasket is set during the creation of the StakingEngine by the EconomyEngineV1Factory, no additional staking baskets can be created using the createStakingBasket() function.

Recommendation

It is recommended to review the logic and change the condition that restricts the creation of new staking baskets based on the defaultStakingBasket value.

18. Anyone can call claim rewards for other user’s item

Severity: Informational	Category: Access Control

Target:
- v3-economy-core\contracts\engines\staking\StakingEngine.sol

Description

In StakingEngine.sol, the claimRewards() function allows msg.sender to claim rewards to another user who staked itemID. It could lead to situations where users receive rewards when they do not wish to do so.

v3-economy-core\contracts\engines\staking\StakingEngine.sol:L534-L537

```
function claimRewards(address basket, bytes32 itemId) external override
validBasket(basket) {
    /// claim rewards
    _claimRewards(basket, itemId, false);
}
```

v3-economy-core\contracts\engines\staking\StakingEngine.sol:L552-L556

```
function _claimRewards(
    address basket,
    bytes32 itemId,
    bool checkAccruingPeriod
) internal {
    ...
}
```

Recommendation

It is recommended to add a verification step to ensure that the caller of the claimRewards() function is the rightful staker of the itemID.

19. Weak ID generation process

Severity: Informational	Category: Business Logic

Target:
- v2-governance-core\contracts\proposal\ProposalHandler.sol
- v2-governance-core\contracts\products\funds\FundManager.sol

Description

v2-governance-core\contracts\proposal\ProposalHandler.sol:L582-L585

```
function _generateProposalID() internal returns (bytes32 proposalID) {
    totalProposal++;
    proposalID = keccak256(abi.encode(_msgSender(), totalProposal));
}
```

In ProposalHandler.sol, the _generateProposalID() function does not incorporate the chainid parameter into the keccak256 hashing process. As a result, there is a possibility for different chains to have identical proposalIDs, potentially causing confusion among users.

v2-governance-core\contracts\products\funds\FundManager.sol:L1280-L1283

```
    function _newID() private returns (bytes32 fundID) {
        _seed++;
        fundID = keccak256(abi.encodePacked(_seed, address(this)));
    }
```

In FundManager.sol, _newID is used during the creation of a new fund. If super admin decides to be at different chains and deploy fundManager using create2 to have the same addresses at all chains, the fundIDs will be the same at different chains which could confuse users.

Recommendation

It is recommended to add the chainid parameter into the process when generating IDs.

20. Could use safeTransfer()/safeTransferFrom() instead of transfer()/transferFrom()

Severity: Informational	Category: Data Validation
Target: - all	

Description

The ink codebase uses transfer() and transferFrom() functions to transfer ERC20 tokens. In some instances, the return value from transfer/transferFrom is checked, while in some instances, they are not.

The issue is that some tokens could return false from the transfer/safeTransfer to indicate the transfer fails. So the calling contract would not notice the failure if the return value is not checked.

On the other hand, some tokens (like USDT on Ethereum) don't correctly implement the EIP20 standard and their transfer/ transferFrom function returns void instead of a success boolean. For these tokens, using the return value from their transfer/transferFrom as bool would cause the execution to revert.

The SafeERC20 library from OpenZeppelin Contract supports both the return-false-on-fail tokens and the return-void-and-revert-instead tokens. It’s recommended to use the safeTransfer()/safeTransferFrom() to replace the use of transfer()/transferFrom().

Recommendation

Consider using safeTransfer()/safeTransferFrom() instead of transfer()/transferFrom().

21. Unused state variables

Severity: Informational	Category: Redundancy
Target: -	

Description

In several contracts, there are state variables that are declared but appear to be unused within the contract. Here are the contracts and the corresponding unused state variables:

- TheBoard.sol:L20

  uint256 private _minPledgeRequired;
- TheBoardV2:L19
  uint256 private _minPledgeRequired;

- BaseDAO:L108,L140,L173

  string private _describe;
  string private _verifyBaseType;
  EnumerableSet.AddressSet private _ucvSet;

- BaseCommittee.sol:L49,L50

  string private _describe;
  bytes[] private _mds;

- BaseUCV.sol:L41,L42

  uint256 private _defaultFlowIDIndex = 0;
  mapping(address => EnumerableSet.UintSet) private ownedNFTs;

- PayrollUCV.sol:L42

  mapping(address => EnumerableSet.UintSet) private ownedNFTs;

- InkFund.sol:L54,L80

  uint256 private _fundAvailablePrincipal = 0;
  uint256 private _voucherValue = 0;

- InvestmentUCVManager.sol:L32

  address private _ucv;

- TreasuryIncomeManager.sol:L31,L35

  uint256 private _proposalID;
  mapping(uint256 => uint256) committedReport;

Recommendation

Review the code to ensure that these variables are indeed not needed for the intended functionality of the contracts. If they are not necessary, consider removing these unused state variables.

22. Typo

Severity: Informational	Category: Typo

Target:
- v2-governance-core\contracts\bases\BaseDAO.sol
- v2-governance-core\contracts\products\funds\FundManager.sol

Description

In various parts of the contracts, there are typographical errors and they can affect readability and maintainability. Here are the typo errors:

- BaseDAO.sol

  IFPS should be IPFS (L73)
  remmoveDuty should be removeDuty (L847)
  propsoals should be proposals (L42)
  functins should be functions (L252)
  economyEngineFactcory should be economyEngineFactory (L599)
  stakinInfo should be stakingInfo (L85)

- FundManager.sol

  _getCommitteeMemeberList should be _getCommitteeMemberList (L243)

Recommendation

Consider correcting the typo as suggestions.

23. Redundant Code

Severity: Informational	Category: Redundancy

Target:
- v2-governance-core\contracts\proposal\ProposalHandler.sol
- v2-governance-core\contracts\committee\TheBoard.sol
- v2-governance-core\contracts\products\funds\FundManager.sol
- v2-governance-core\contracts\agents\InvestmentManagementSetupAgent.sol
- v2-governance-core\contracts\bases\BaseDAO.sol
- v2-governance-core\contracts\cores\IdentityManager.sol

Description

v2-governance-core\contracts\proposal\ProposalHandler.sol:L491-L495

```
function syncProposalKvDataToTopic(
    bytes32 proposalID,
    bool agree,
    bytes memory
) internal {
    ...
    if (p.topicID == bytes32(0x0)) {
        topicID = keccak256(abi.encode(proposalID));
    } else {
        topicID = p.topicID;
    }
    ...
}
```

In the decideProposal function, syncProposalKvDataToTopic() is called. It checks the provided topicID of the proposal and generates an ID if topicID is 0x0. However, it seems that the initial topicID check is already done when creating a proposal.

v2-governance-core\contracts\proposal\ProposalHandler.sol:L142-L161

```
function _getProposalFlow(bytes32 proposalID)
    internal
    view
    returns (bytes32 flowID)
{
    bytes32 proposalFlowID =
0x0000000000000000000000000000000000000000000000000000000000000004;
    ...
    if (
        proposalFlowID ==
        0x0000000000000000000000000000000000000000000000000000000000000004
    ) {
        flowID = _defaultFlows[_defaultFlowIDIndex];
    } else {
        ...
    }
}
```

There is a condition check on proposalFlowID in the _getProposalFlow() function. However, it seems that the assignment of proposalFlowID to 0x0000000000000000000000000000000000000000000000000000000000000004 at the beginning of the function makes this condition check meaningless. It might be a logical error,

as the code within the else branch will never be executed. It's recommended to review the logic concerning proposalFlowID in the _getProposalFlow() function and make the necessary adjustments.

v2-governance-core\contracts\proposal\ProposalHandler.sol:L830

```
function _execFinish(ProposalProgress storage info, bool agree) internal {
    ...
    if (agree == false) {
        ...
    }
    ...
    if (agree == true) {
        ...
    }
    ...
}
```

In the _execFinish function, the conditional statement at line 830 is redundant.

v2-governance-core\contracts\committee\TheBoard.sol:L225-L232

```
function _tallyVotes(VoteIdentity memory identity, bytes memory data)
    internal
{
    ...
    bool passOrNot = _calculateVoteResults(identity, true, basePassSeat);

    VoteInfo storage voteInfo = _voteInfos[identity._getIdentityID()];
    if (passOrNot) {
        voteInfo.status = VoteStatus.AGREE;
    } else {
        voteInfo.status = VoteStatus.DENY;
    }
    ...
}
```

There have been modifications to the voteInfo.status in the _calculateVoteResults(), but when the function call returns to _tallyVotes(), the same operation is performed again.

v2-governance-core\contracts\products\funds\FundManager.sol:L336-L340

```
function _addIntoWhitelist(
    bytes32 fundID,
    WhitelistToken memory wl,
    int256 defaultApproved
) internal {
    ...
    if (defaultApproved == 1) {
        token.auditApproved = defaultApproved;
    } else {
        token.auditApproved = 1;
    }
    ...
}
```

The token.auditApproved can be directly assigned the value of 1 instead of being checked using an if-else statement.

v2-governance-core\contracts\agents\InvestmentManagementSetupAgent.sol:L106-L110

```
function exec(bytes32 proposalID) external override onlyCallFromDAO {
     ...
     address ucvManager = IDAO(getAgentDAO()).deployByKey(
```

```
            typeID,
            bytesData.toBytes32(),
            abi.encode(proposalID)
        );

        _executed = true;
    }
```

The ucvManager in the exec() function is not used.

v2-governance-core\contracts\bases\BaseDAO.sol:L1338-L1344

```
function setupFlowInfo(FlowInfo memory flow) external override onlyAgent {
    _setFlowStep(flow);
    }

function setFlowStep(FlowInfo memory flow) external override onlyAgent {
    _setFlowStep(flow);
}
```

The setupFlowInfo() and the setFlowStep() functions are identical. It is suggested to delete one of them.

v2-governance-core\contracts\cores\IdentityManager.sol:L31

```
function init(
    address owner,
    address config,
    bytes calldata data
) external override initializer returns (bytes memory callback) {
    ...
}
```

The config param is not used in the init() function. The issue can also be found in the init() functions of the InkBadgeERC20, KYCVerifyManager, and IdentityManagerV2 contracts.

Recommendation

Consider removing the redundant code as suggestions.

24. Inconsistent function name and logic in canVoteChangeResult()

Severity: Informational	Category: Business Logic

Target:
- v2-governance-core\contracts\committee\TheBoard.sol

Description

At the end of the vote() function, it will be judged whether this vote can change the voting result, and based on this, it will be decided whether to execute the _tallyVote() function. According to the function name canVoteChangeResult(), we can infer that the function of this function is to determine whether this vote will affect the status of voting on the proposal, when true is returned, it means it will be affected, and vice versa. However, in the code implementation of this function, the function returns false for conditions that may affect the voting results, otherwise it returns true.

v2-governance-core/contracts/committee/TheBoard.sol:L135-L145

```
function canVoteChangeResult(
    uint256 agreeVotes,
    uint256 disagreeVotes,
    uint256 baseAgreeSeat
) public view returns (bool) {
    if (agreeVotes >= baseAgreeSeat || disagreeVotes >= baseAgreeSeat) {
        return false;
    }

    return true;
}
```

Recommendation

It is recommended to align the function name with its logic.

Appendix

Appendix 1 - Audit Scope

We performed a diff-audit of:

1. the Ink-Finance-Inc/v2-governance-core repository at commit 2e9841f against the version of the same repository at commit 193120f. In scope were the following contracts:

File	SHA-1 hash
contracts/agents/DAOInfoSetupAgent.sol	6abad5917d41d7d07ad0a0bd2150ff06180dd0f1
contracts/agents/InvestmentManagementSetupAgent.sol	3843a4e07db6bf1eeb84dfc8cfad53b2bb6da45b
contracts/agents/TreasuryManagerAgent.sol	7d744e6ad3c4cc0cef50b7a548c3349b6f854d5f
contracts/bases/BaseAgent.sol	e68c194d79918a61460b9111e47bf141e47ea06e
contracts/bases/BaseCommittee.sol	51d963d21b27b1a78d63173e2d4b53cb38dffbd0
contracts/bases/BaseDAO.sol	c71c6797038515f9848477abe2dd06c5ad442a74
contracts/bases/BaseUCV.sol	4cb6f0346d55f7915a0a74298531ec3fe3e1ddfb
contracts/committee/InvestmentCommittee.sol	3d74dfbc7e6a78814dbf86db1ea0a04017fe56a4
contracts/committee/TheBoard.sol	fa455e00a5311877d09d97209d885e882310c99c
contracts/committee/TheBoardV2.sol	2c1823befa67e1bd1f07e9b6f0b40f9051248597
contracts/committee/ThePublic.sol	f7e80e5615466947dafe84f064f5af18dfb7827b
contracts/committee/TreasuryCommittee.sol	568a86c98c86805dc0e5b419c9b8eee4437f7865
contracts/cores/IdentityManager.sol	2e26012b23a484de545c22f34c9fa9a89485307d
contracts/cores/IdentityManagerV2.sol	be40dbc2a1c238eec32ed9cc4e5fca9ef2c25d24
contracts/cores/KYCVerifyManager.sol	3624e2afe848c9df7c5b0da1a11596510caa85b0
contracts/daos/MasterDAO.sol	806d60ff76dc878a5acd5e529b4c8cb916384a65
contracts/products/EscrowManager.sol	12137b50c89b82188d299c839e5c03c93eeff40b
contracts/products/InkEnvelopeFactory.sol	29e5941b8c4957498d0cdaeba30242821487993a
contracts/products/InkFund.sol	2264b6b2e5dc4f3b1b567b39b31a0b8c5217709b
contracts/products/TradeRouter.sol	4bc7210be79633b33e215d8b8db3a082bb93cd5b
contracts/products/funds/FundManager.sol	862cd18819ebc1bf5dbf453ca6508f1687bcc51d
contracts/products/tokens/WrappedERC1155.sol	dd973ceabfe2017580b959f55b2a774eebddcec6
contracts/products/tokens/WrappedERC20.sol	7790b94072528c3ce4c6762781c6bd0e4a7c0930
contracts/products/tokens/WrappedERC721.sol	164e92d651c0c06f2e8f6426ae1f7c94f5ca2399
contracts/products/tokens/WrappedToken.sol	d159423184d0ef94284ebad546b6d1f4e181708e
contracts/proposal/ProposalHandler.sol	013520ba782c75f1255980b8d8c404f40e4f972b
contracts/tokens/InkBadgeERC20.sol	01898a5a7cbb76d2eb2d048ac87541a7db74589f
contracts/tokens/InkERC20.sol	8ea58289bcd994f53e5c4b8e9b25dc6df71786d7
contracts/tokens/InkERC721.sol	7a749ef4d5392dc7324020d1f89dbd39f713be77
contracts/tokens/InkFundCertificateToken.sol	3057508cca98f8f409bf08225f5b2ccd125e35f8

File	SHA-1 hash
contracts/ucv/InvestmentUCVManager.sol	029f0c5f31fa550634a46317c87f75d06d269952
contracts/ucv/PayrollUCV.sol	5b23cfeadc06b354c0a065cff27e7d7b989af66c
contracts/ucv/PayrollUCVManager.sol	b3c6b1ebec2f430cf70291509316ad59a6242730
contracts/ucv/TreasuryIncomeManager.sol	e3ccbed83012721132dd6109f452b36eb61dcf0e
contracts/upgrade/FactoryManager.sol	4b86d4ca556eff2169493e01fa12801423d8f2b5
contracts/upgrade/InkBeaconProxy.sol	775b63c20a817adc7bb507214ccff21ec6b18373
contracts/utils/ConfigManager.sol	aa6521c10bdaa012f57097ba421be4927e36f53c

2. the Ink-Finance-Inc/v3-economy-core repository at commit 799f3bb against the version of the same repository at commit 0c62e9c. In scope were the following contracts:

File	SHA-1 hash
contracts/EconomyEngineV1Factory.sol	67d52ddd562a68a4652380575fab1ea993021d28
contracts/config/EconomyAddressesProvider.sol	1e89420ef6f2e10b3fef5aa793e6337ea4670162
contracts/engines/pledge/PledgeEngine.sol	46891ca5f5149f12ebe1c5269063ce2d21ab9393
contracts/engines/pools/BasketPool.sol	5c359d53e9e74a110bb95f9d44a4446bd63682b4
contracts/engines/pools/EmissionPool.sol	35add071a541d0bf96f624af4ea31bdbf06d7dc7
contracts/engines/pools/PrinciplePool.sol	9bc5e9fe5181006d00e705da3e0c363d2b4c7db4
contracts/engines/sponsor/InkMainSponsorEngine.sol	942f155c7a8e4938ebf53aaf0725b86fc5ab44f2
contracts/engines/sponsor/SponsorEngine.sol	89d679b24156de5676445d8ed70805da33cb0cd0
contracts/engines/sponsor/SponsorEngineV2.sol	a466c9d4c278933ce430019d49a8e4800c525ffe
contracts/engines/staking/InkMainStakingEngine.sol	3a039beb789c0210dcf97102e76330f6f880380c
contracts/engines/staking/StakingBasket.sol	97a8da0fb54cb74adf4df89d6c4eedc3aa890781
contracts/engines/staking/StakingEngine.sol	2e1dc1627f59d64ff42ee9cd06d822d67cf0add9
contracts/engines/trust/TrustEngine.sol	1bc4b01afb2e6c7eb8bffa3b2b761a44a739c1b1
contracts/interfaces/IBasketPool.sol	a0390896ee9a856f495d6392c022808802a1a9c9
contracts/interfaces/IBufferPool.sol	a57ff7e09870715fe9d653328d9300653cb175d3
contracts/interfaces/IDaoGovernance.sol	1edd52eb838f6d0173438745ab7b4814606e9eaf
contracts/interfaces/IEconomyAddressesProvider.sol	396de7a3ba4f8b67452845aec27fa7b85d0f29c5
contracts/interfaces/IEconomyEngineV1Factory.sol	86aaa199fb454373b9bdaedf27b4963736dadbcb
contracts/interfaces/IEmissionPool.sol	5af547aff66e45d9594b1937aa3b478949e0760c
contracts/interfaces/IInkMainStakingEngine.sol	d0b04ad52e576f1299f85299ae5e7b18f280ce17
contracts/interfaces/IPledgeEngine.sol	6de92a5caf9b7f1546110b8b874cbcec83c1256d
contracts/interfaces/IPrinciplePool.sol	2c2b2618753ac42219d4f352d7711ee6872b865a
contracts/interfaces/IReclaimPool.sol	24ce328338ddac83853b3a6bfba2bda6ee0f98a3
contracts/interfaces/ISponsorEngine.sol	75a233b9c9b739969c44945d84224b4d51dd47ea
contracts/interfaces/IStakingBasket.sol	3f02ab17d9beb8acaa499aa45155d1330e314932
contracts/interfaces/IStakingEngine.sol	19bb8624613a3e78819f8f96ab2c6b1978d0b01b

File	SHA-1 hash
contracts/libraries/LPledgeEngine.sol	95dafd28aeab8d65b8287fe8c8b3aea73c6771c4
contracts/libraries/LSponsorEngine.sol	200f2ad6559a247146ec8b10c21eeaa7d28b17b2
contracts/libraries/LStakingBasket.sol	9ea7f84b5cf12e3c721c4d11bae7d6d6fa26d691
contracts/libraries/LStakingEngine.sol	a89185f1da97fb45456f782fb2d94a868d6ebb95
contracts/libraries/LStakingItem.sol	7dd5dcabf989ff5b51a626aa942705b24022cad6
contracts/libraries/PRBMath.sol	7dd5dcabf989ff5b51a626aa942705b24022cad6
contracts/libraries/PRBMathUD60x18.sol	7455bf2429527145446f2a3dd0e9ffe8ff182cc5