$

Competitive Security Assessment

Influpia

Mar 19th, 2024

Secure3

secure3.io

Summary

This report is prepared for the project to identify vulnerabilities and issues in the smart contract source code. A group of NDA covered experienced security experts have participated in the Secure3's Audit Contest to find vulnerabilities and optimizations. Secure3 team has participated in the contest process as well to provide extra auditing coverage and scrutiny of the finding submissions.

The comprehensive examination and auditing scope includes:

• Cross checking contract implementation against functionalities described in the documents and white paper disclosed by the project owner.

• Contract Privilege Role Review to provide more clarity on smart contract roles and privilege.

• Using static analysis tools to analyze smart contracts against common known vulnerabilities patterns.

• Verify the code base is compliant with the most up-to-date industry standards and security best practices.

• Comprehensive line-by-line manual code review of the entire codebase by industry experts.

The security assessment resulted in findings that are categorized in four severity levels: Critical, Medium, Low, Informational. For each of the findings, the report has included recommendations of fix or mitigation for security and best practices.

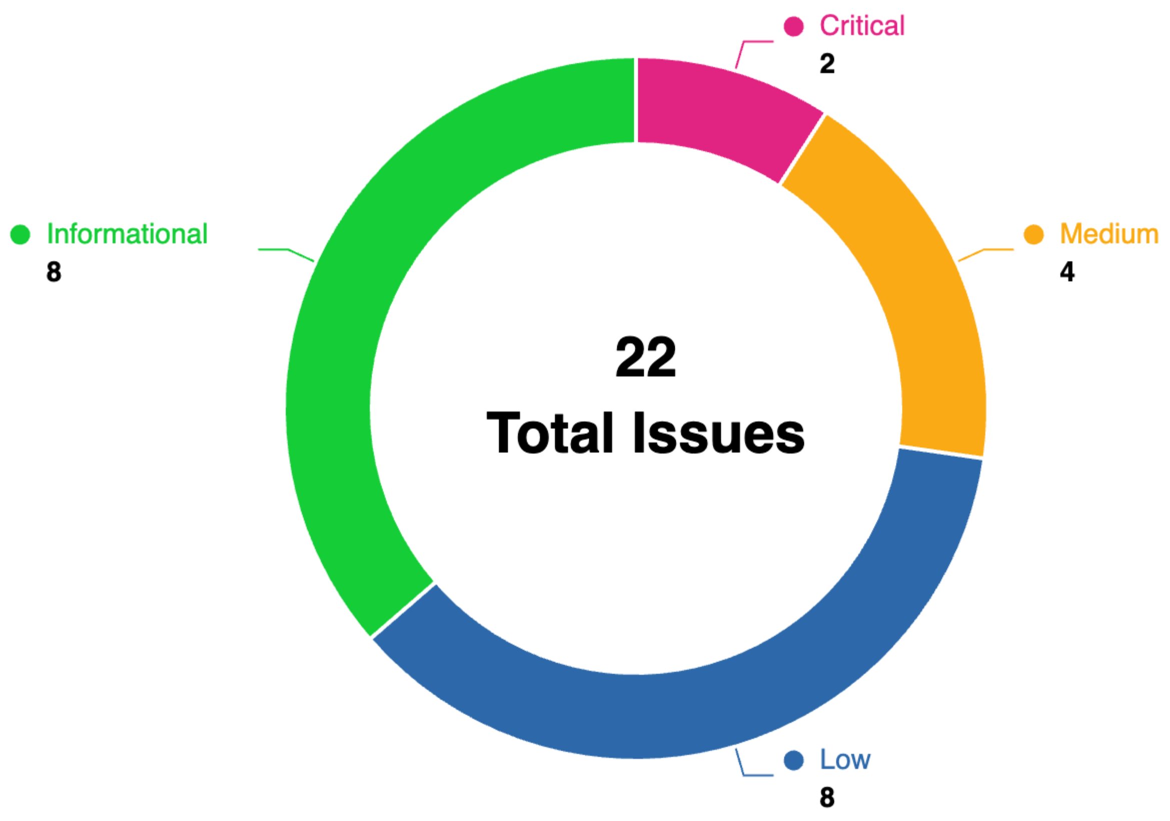
Overview

Project Name	Influpia
Language	Solidity
Codebase	• https://github.com/influpia-aduit/influpia-contract • audit version - 3d006d7168e5039c1cdea70f56cc7e2220df0463 • final version - c1f5a97f13e0edbc535a5f6a307e7335148069d4
Audit Methodology	• Audit Contest • Business Logic and Code Review • Privileged Roles Review • Static Analysis

Audit Scope

File	SHA256 Hash
./src/SharePoolV3.sol	85656ff26edb015cce3b1f314e199c8793711620c9e10863edafd852373930d4
./src/InflupiaMarket.sol	bb241bdac055508f890a7dee90a1fcaa8335899945ffc261722cd1a0c7c6cc5a
./src/InflupiaChef.sol	489340787d0fc538982c116ad90dcd0bacde1371ea2d3caa64e00015a3077f6c
./src/InflupiaCompetition.sol	d4fc2e3617ffdc4665b8f51389c4cb7788020fd54c14e9ec4a720991f03be096
./src/Influpia.sol	90844d134fd30d5da21aead5ec5eafad2f5c5087b7d99006ab469137f1242c0d

Code Assessment Findings



ID	Name	Category	Severity	Client Response	Contributor
IFP-1	totalDeposits is not updated in withdraw function causing the incorrect calculation for rewards	Logical	Critical	Fixed	biakia
IFP-2	The new competition round does not clear maxPower and latestTopTrader, causing the winner of the previous round to also win this round.	Logical	Critical	Declined	thereksfour
IFP-3	Influpia.sol contract is vulnerable to signature malleability	Signature Forgery or Replay	Critical	Fixed	grep-er
IFP-4	discount() does not update totalHoldWorth	Logical	Medium	Fixed	thereksfour
IFP-5	`hostInfo` will not be updated when a user changes his host	Logical	Medium	Fixed	biakia, thereksfour
IFP-6	Weak Sources of Randomness	Weak Sources of Randomness	Medium	Declined	0xac, biakia

IFP-7	Lack of check for `totalBP` in function `setFeeBP`	Logical	Medium	Fixed	0xac, biakia
IFP-8	Centralized Risk With Coin Mint in src/influpia.sol	Privilege Related	Medium	Fixed	Xi_Zi
IFP-9	onlyTradeStarted should use block.number instead of block.timestamp	DOS	Low	Declined	grep-er, ravikiran_web3, thereksfour
IFP-10	`setCompetitionStart` Code Logic Verification	Logical	Low	Acknowledged	Xi_Zi
IFP-11	The `estimateSecondReward` function may always return 0	Logical	Low	Acknowledged	biakia
IFP-12	The `discount` function should limit the minimum value of the price	Logical	Low	Acknowledged	0xac
IFP-13	Ownership change should use two-step process	Privilege Related	Low	Acknowledged	biakia
IFP-14	Off by one error on InflupiaMarket::_safeCastTo128()	Integer Overflow and Underflow	Low	Acknowledged	grep-er
IFP-15	InflupiaCompetition::execute() does not check status of transfer ether function	Logical	Low	Acknowledged	0xac, ravikiran_web3
IFP-16	Expensive irrelevant code in InflupiaMarket::setFeeBP()	Language Specific	Low	Acknowledged	grep-er
IFP-17	Competition rewards may be transferred to `address(0)`	Logical	Low	Acknowledged	biakia
IFP-18	These set functions need to record events	Code Style	Informational	Acknowledged	0xac
IFP-19	Some of the rewards will be locked in `SharePoolV3`	Logical	Informational	Acknowledged	biakia
IFP-20	Rounding Error in `InflupiaChef:_claim()` function	Logical	Informational	Acknowledged	newway55
IFP-21	Remove unused imports	Code Style	Informational	Acknowledged	biakia
IFP-22	Missing error message in require statement	Code Style	Informational	Acknowledged	biakia

IFP-23	Missing Event Setter in contracts	Language Specific	Informational	Acknowledged	Xi_Zi
IFP-24	Meaningful values are hardcoded	Code Style	Informational	Acknowledged	biakia
IFP-25	It is recommended to use different variable names to distinguish `UserInfo.amount` and `minerInfo.power`	Code Style	Informational	Acknowledged	0xac

IFP-1:totalDeposits is not updated in withdraw function causing the incorrect calculation for rewards

Category	Severity	Client Response	Contributor
Logical	Critical	Fixed	biakia

Code Reference

- code/src/InflupiaChef.sol#L98-L106

```
98: function withdraw(address acct, uint256 amount) external {
99:         if (msg.sender != agent) revert INVALID_CHEF_AGENT();
100:         _updateUser(acct);
101:
102:         address currHost = userInfo[acct].host;
103:         userInfo[acct].amount -= amount;
104:         hostInfo[currHost].amount -= amount;
105:         emit Withraw(acct, amount);
106:     }
```

Description

biakia: In `InflupiaChef` contract, the function `deposit` will increase the `totalDeposits`:

```
function deposit(address acct, address host, uint256 amount) external override {
        if (msg.sender != agent) revert INVALID_CHEF_AGENT();
        _updateUser(acct);

        address currHost = userInfo[acct].host;
        hostInfo[currHost].amount -= userInfo[acct].amount;
        userInfo[acct].host = host;
        userInfo[acct].amount += amount;
        hostInfo[host].amount += userInfo[acct].amount;
        totalDeposits += amount;

        emit Deposit(acct, amount);
    }
```

However, in `withdraw` function, the `totalDeposits` is not decreased:

```
function withdraw(address acct, uint256 amount) external {
        if (msg.sender != agent) revert INVALID_CHEF_AGENT();
        _updateUser(acct);

        address currHost = userInfo[acct].host;
        userInfo[acct].amount -= amount;
        hostInfo[currHost].amount -= amount;
        emit Withraw(acct, amount);
    }
```

The `` `totalDeposits` `` is used in function `` `_update()` `` to calculate the rewards per share:

```
 function _update() private {
        if (block.timestamp <= lastRewardTimestamp || totalDeposits == 0) return;

        unchecked {
            uint256 rewards = (block.timestamp - lastRewardTimestamp) * rewardPerSecond;

            perShareIndex += (rewards * TIMES) / totalDeposits;
        }

        lastRewardTimestamp = block.timestamp;
    }
```

As a result, when a user withdraws his amount, the `` `totalDeposits` `` will not decrease and the `` `perShareIndex` `` will be a little smaller than expected. When more and more users withdraw their amounts, the `` `perShareIndex` `` will be more and more smaller than expected.

Recommendation

biakia: Consider decreasing the `` `totalDeposits` `` in `` `withdraw` `` function:

```
function withdraw(address acct, uint256 amount) external {
        if (msg.sender != agent) revert INVALID_CHEF_AGENT();
        _updateUser(acct);

        address currHost = userInfo[acct].host;
        userInfo[acct].amount -= amount;
        hostInfo[currHost].amount -= amount;
        totalDeposits -= amount;
        emit Withraw(acct, amount);
    }
```

Client Response

biakia: Fixed. withdraw is not used in the whole project, but totalDeposits did forget to deal with it.
fixed: https://github.com/influpia-aduit/influpia-contract/commit/026dd414360e990b3dd7f369d25dcaf5353fb828

IFP-2:The new competition round does not clear maxPower and latestTopTrader, causing the winner of the previous round to also win this round.

Category	Severity	Client Response	Contributor
Logical	Critical	Declined	thereksfour

Code Reference

Description

thereksfour: In InflupiaCompetition, when power > maxPower, new maxPower and latestTopTrader will be set.

```
        if (power > maxPower) {
            maxPower = power;
            bool topTraderChanged = (latestTopTrader != trader);
            latestTopTrader = trader;
            if (topTraderChanged &&
                competitions[competitionRound].status == STATUS_STARTED &&
                block.timestamp > uint256(competitions[competitionRound].expectingWinTime - compet
itions[competitionRound].winGapTime)
            ) {
                competitions[competitionRound].expectingWinTime = uint64(block.timestamp + competi
tions[competitionRound].winGapTime);
            }
        }
```

After the end of this round of competition, latestTopTrader will be the winner and receive the reward.

```
        if (competitions[competitionRound].status == STATUS_STARTED &&
            block.timestamp > uint256(competitions[competitionRound].expectingWinTime)
        ) {
            competitions[competitionRound].status = STATUS_ENDED;
            uint256 competitionReward = address(this).balance;
            competitions[competitionRound].totalReward = competitionReward * competitions[competit
ionRound].rewardBP / BP;
            _transferETH(latestTopTrader, competitions[competitionRound].totalReward);
            emit CompetitionEnded(competitionRound, competitions[competitionRound].totalReward, la
testTopTrader);
        }
```

The problem here is that `maxPower` and `latestTopTrader` are common to all competition rounds, and when setCompetitionStart sets a new competition round, maxPower and latestTopTrader are not cleared.

```
    function setCompetitionStart(
        uint64 startTime,
        uint64 firstEndingGap,
        uint64 winGapTime,
        uint32 rewardBP
    ) external onlyOwnerOrOperator {
        require(uint256(startTime) > block.timestamp, "invalid time");
        if (competitions[competitionRound].status == STATUS_ENDED) {
            competitionRound++;
        }
        require(competitions[competitionRound].status == STATUS_NOT_EXISTS, "cannot start new comp
etition");
        require(rewardBP <= BP, "invalid rewardBP");
        competitions[competitionRound].startTime = startTime;
        competitions[competitionRound].expectingWinTime = startTime + firstEndingGap;
        competitions[competitionRound].winGapTime = winGapTime;
        competitions[competitionRound].status = STATUS_NOT_STARTED;
        competitions[competitionRound].rewardBP = rewardBP;

    }
```

This results in that in a new competition round, unless a user's power exceeds the maxPower of the winner of the previous round, the winner of the previous round will still be the winner of this round and will receive the reward.

Recommendation

thereksfour: It is recommended to clear maxPower and latestTopTrader in setCompetitionStart

```
    function setCompetitionStart(
        uint64 startTime,
        uint64 firstEndingGap,
        uint64 winGapTime,
        uint32 rewardBP
    ) external onlyOwnerOrOperator {
        require(uint256(startTime) > block.timestamp, "invalid time");
        if (competitions[competitionRound].status == STATUS_ENDED) {
            competitionRound++;
        }
        require(competitions[competitionRound].status == STATUS_NOT_EXISTS, "cannot start new comp
etition");
        require(rewardBP <= BP, "invalid rewardBP");
        competitions[competitionRound].startTime = startTime;
        competitions[competitionRound].expectingWinTime = startTime + firstEndingGap;
        competitions[competitionRound].winGapTime = winGapTime;
        competitions[competitionRound].status = STATUS_NOT_STARTED;
        competitions[competitionRound].rewardBP = rewardBP;
+       maxPower = 0;
+       latestTopTrader = address(0);
    }
```

Client Response

thereksfour: Declined. By design

IFP-3:Influpia.sol contract is vulnerable to signature malleability

Category	Severity	Client Response	Contributor
Signature Forgery or Replay	Critical	Fixed	grep-er

Code Reference

- code/src/ERC404.sol#L297

```
297: function permit(
```

- code/src/Influpia.sol#L10

```
10: contract Influpia is ERC404, Ownable {
```

Description

grep-er: Influpia inherits ERC404 which uses permit function which is vulnerable to signature malleability as it doesn't check values of `s` in v,r,s

```
contract Influpia is ERC404, Ownable {
```

In future permit function may cause problem when using it
ERC404.sol

```
  function permit(
    address owner_,
    address spender_,
    uint256 value_,
    uint256 deadline_,
    uint8 v_,
    bytes32 r_,
    bytes32 s_
  ) public virtual {// @audit signature maleblity but not inscope but is is used in infulpia mark
t.sol write that it is not diriiectly so med but in future
    if (deadline_ < block.timestamp) {
      revert PermitDeadlineExpired();
    }

    if (value_ <= _minted && value_ > 0) {
      revert InvalidApproval();
    }

    if (spender_ == address(0)) {
      revert InvalidSpender();
    }

    unchecked {
      address recoveredAddress = ecrecover(
        keccak256(
          abi.encodePacked(
            "\x19\x01",
            DOMAIN_SEPARATOR(),
            keccak256(
              abi.encode(
                keccak256(
                  "Permit(address owner,address spender,uint256 value,uint256 nonce,uint256 deadli
ne)"
                ),
                owner_,
                spender_,
                value_,
                nonces[owner_]++,
                deadline_
              )
            )
          )
        ),
        v_,
        r_,
        s_
      );
```

Recommendation

grep-er: Add check to not allow s value more then `0x7FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5D576E7357A4501DDFE92F46681B20A0`

```
function permit(
    address owner_,
    address spender_,
    uint256 value_,
    uint256 deadline_,
    uint8 v_,
    bytes32 r_,
    bytes32 s_
  ) public virtual {// @audit signature maleblity but not inscope but is is used in infulpia mark
t.sol write that it is not diriiectly so med but in future
    if (deadline_ < block.timestamp) {
      revert PermitDeadlineExpired();
    }
++         if (uint256(s) > 0x7FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5D576E7357A4501DDFE92F46681B20A0) re
vert();
```

or
Use OpenZeppelin ecdsa
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/cryptography/ECDSA.sol#L137

Client Response

grep-er: Fixed. fixed: https://github.com/influpia-aduit/influpia-contract/commit/ebf96ccb08d44f3d3bfb118f040599182c824f6d

IFP-4:discount() does not update totalHoldWorth

Category	Severity	Client Response	Contributor
Logical	Medium	Fixed	thereksfour

Code Reference

- code/src/InflupiaMarket.sol#L145-L155

```
function discount(address miner, uint256 worth) external onlyTradeStarted {
        Miner memory m = miners[miner];

        if (m.host == address(0)) revert MINNER_NOT_EXIST();
        if (m.host != msg.sender) revert UNAUTHORIZED();

        if (worth == 0 || worth >= m.worth) revert INVALID_WORTH();
        miners[miner].worth = _safeCastTo128(worth);

        emit Discount(miner, worth);
    }
```

Description

thereksfour: The host can call discount to reduce miners.worth.

```
    function discount(address miner, uint256 worth) external onlyTradeStarted {
        Miner memory m = miners[miner];

        if (m.host == address(0)) revert MINNER_NOT_EXIST();
        if (m.host != msg.sender) revert UNAUTHORIZED();

        if (worth == 0 || worth >= m.worth) revert INVALID_WORTH();
        miners[miner].worth = _safeCastTo128(worth);

        emit Discount(miner, worth);
    }
```

However as _grab does, when miners.worth is changed, totalHoldWorth also needs to be changed.

```
    function _grab(address miner) private {
        Miner memory m = miners[miner];

        if (m.worth != msg.value) revert INVALID_BUY_PRICE();
        if (m.host == msg.sender) revert INVALID_BUYER();
        influpiaChef.deposit(miner, msg.sender, _growthPower(msg.value, 0));
        totalHoldWorth[miners[miner].host] -= msg.value;
        uint128 nextWorth = _safeCastTo128(getNextWorth(msg.value));
        totalHoldWorth[msg.sender] += nextWorth;
        miners[miner].host = msg.sender;
        miners[miner].worth = nextWorth;
```

This will cause totalHoldWorth to be incorrect, affecting subsequent integrations.

Recommendation

thereksfour: It is recommended to change totalHoldWorth in discount

```
    function discount(address miner, uint256 worth) external onlyTradeStarted {
        Miner memory m = miners[miner];

        if (m.host == address(0)) revert MINNER_NOT_EXIST();
        if (m.host != msg.sender) revert UNAUTHORIZED();

        if (worth == 0 || worth >= m.worth) revert INVALID_WORTH();
+       totalHoldWorth[msg.sender] -= m.worth;
+       totalHoldWorth[msg.sender] += worth;
        miners[miner].worth = _safeCastTo128(worth);

        emit Discount(miner, worth);
    }
```

Client Response

thereksfour: Fixed.fixed: https://github.com/influpia-aduit/influpia-contract/commit/97cbe16d454ef893a5ec3ccfa22e63755296b40b

IFP-5: `hostInfo` will not be updated when a user changes his host

Category	Severity	Client Response	Contributor
Logical	Medium	Fixed	biakia, thereksfour

Code Reference

- code/src/InflupiaChef.sol#L108-L111
- code/src/InflupiaChef.sol#L108-L111

```
108: function setHost(address acct, address host) external override {
109:         if (msg.sender != agent) revert INVALID_CHEF_AGENT();
110:         userInfo[acct].host = host;
111:     }
```

```
108: function setHost(address acct, address host) external override {
109:         if (msg.sender != agent) revert INVALID_CHEF_AGENT();
110:         userInfo[acct].host = host;
111:     }
```

Description

biakia: In contract `` `InflupiaChef` ``, the function `` `setHost` `` is used to change the user's host:

```
function setHost(address acct, address host) external override {
        if (msg.sender != agent) revert INVALID_CHEF_AGENT();
        userInfo[acct].host = host;
    }
```

The issue here is that the `` `hostInfo` `` does not be updated.
thereksfour: The agent can call setHost to change userInfo.host.

```
    function setHost(address acct, address host) external override {
        if (msg.sender != agent) revert INVALID_CHEF_AGENT();
        userInfo[acct].host = host;
    }
```

However, as deposit does, when userInfo.host is changed, hostInfo.amount also needs to be changed.

```
    function deposit(address acct, address host, uint256 amount) external override {
        if (msg.sender != agent) revert INVALID_CHEF_AGENT();
        _updateUser(acct);

        address currHost = userInfo[acct].host;
        hostInfo[currHost].amount -= userInfo[acct].amount;
        userInfo[acct].host = host;
        userInfo[acct].amount += amount;
        hostInfo[host].amount += userInfo[acct].amount;
        totalDeposits += amount;

        emit Deposit(acct, amount);
    }
```

This will cause hostInfo.amount to be incorrect, potentially causing withdraw to revert due to underflow

```
    function withdraw(address acct, uint256 amount) external {
        if (msg.sender != agent) revert INVALID_CHEF_AGENT();
        _updateUser(acct);

        address currHost = userInfo[acct].host;
        userInfo[acct].amount -= amount;
        hostInfo[currHost].amount -= amount;
        emit Withraw(acct, amount);
    }
```

Recommendation

biakia: Consider updating the data of `hostInfo` when the user's host has been changed:

```
function setHost(address acct, address host) external override {
        if (msg.sender != agent) revert INVALID_CHEF_AGENT();
        address currHost = userInfo[acct].host;
        hostInfo[currHost].amount -= userInfo[acct].amount;
        userInfo[acct].host = host;
        hostInfo[host].amount += userInfo[acct].amount;
    }
```

thereksfour:

```
It is recommended to change hostInfo.amount in setHost
```

```
    function setHost(address acct, address host) external override {
        if (msg.sender != agent) revert INVALID_CHEF_AGENT();
+       address currHost = userInfo[acct].host;
+       hostInfo[currHost].amount -= userInfo[acct].amount;
        userInfo[acct].host = host;
+       hostInfo[host].amount += userInfo[acct].amount;
    }
```

Client Response

biakia: Fixed.setHost is not used by market any longer, but it is really incorrect.
fixed: https://github.com/influpia-aduit/influpia-contract/commit/2f684fce0b188fd522a1a106aa3e13b3e598a304
thereksfour: Fixed.setHost is not used by market any longer, but it is really incorrect.
fixed: https://github.com/influpia-aduit/influpia-contract/commit/2f684fce0b188fd522a1a106aa3e13b3e598a304

IFP-6:Weak Sources of Randomness

Category	Severity	Client Response	Contributor
Weak Sources of Randomness	Medium	Declined	0xac, biakia

Code Reference

Description

0xac: The generation factors of `` `randomRaw` `` parameters (`` `from_, to_, block.timestamp, balance, id_` ``) can be known in advance, which allows the attacker to control certain parameters (such as `` `from_, to_, id_` ``) to construct `` `randomRaw ` `` The value of the parameter has achieved the purpose of controlling the value of `` `attribute.value` ``.

```
uint256 randomRaw = uint256(keccak256(abi.encodePacked(from_, to_, block.timestamp, balance, id
_)));
```

biakia: The function `` `_transferERC721` `` is using `` `uint256(keccak256(abi.encodePacked(from_, to_, block.timestamp, balance, id_)))` `` to get a random number. However, this random number is totally calculated on-chain, which is exploitable. The attacker can transfer some tokens to the address `` `0x453EFb70b21f9E4a37f7B181a99d63817D0313d1` `` to manipulate the `` `balance` `` and finally manipulate the `` `probability` ``.

Recommendation

0xac: It is recommended to change the unpredictable random number generation method or use other methods to generate `` `attribute.value` ``.
biakia: Consider using chainlink's VRF to generate a safe random number.

Client Response

0xac: Declined. According to the ERC404 protocol, a new ERC721 will be generated at the time of transfer, and when the new NFT has no attributes set, we will compute the random value based on the current transaction information. vrf needs to be executed asynchronously, which is not operable for normal ERC20 transfers, and in order to make it more difficult for the random value to be predicted, we have introduced merlinswap To increase the difficulty of predicting the random value, we introduce the number of WBTCs in the most frequently traded VOYA/WBTC pool as part of the random seed, which varies with the moment of the transaction. Second, direct user transactions also affect each other. When a user's balance is smaller than unit, the NFT will enter the recycling queue, and other users will prefer to get the NFT from the recycling queue when they get a new NFT, which will further increase the difficulty of predicting the chance of success of the random value.
biakia: Declined. According to the ERC404 protocol, a new ERC721 will be generated at the time of transfer, and when the new NFT has no attributes set, we will compute the random value based on the current transaction information. vrf needs to be executed asynchronously, which is not operable for normal ERC20 transfers, and in order to make it more difficult for the random value to be predicted, we have introduced merlinswap To increase the difficulty of predicting the random value, we introduce the number of WBTCs in the most frequently traded VOYA/WBTC pool as part of the random seed, which varies with the moment of the transaction. Second, direct user transactions also affect each other. When a user's balance is smaller than unit, the NFT will enter the recycling queue, and other users will prefer to get the NFT from the recycling queue when they get a new NFT, which will further increase the difficulty of predicting the chance of success of the random value.

IFP-7:Lack of check for `totalBP` in function `setFeeBP`

Category	Severity	Client Response	Contributor
Logical	Medium	Fixed	0xac, biakia

Code Reference

- code/src/InflupiaMarket.sol#L212-L223
- code/src/InflupiaMarket.sol#L212-L223

```
212: function setFeeBP(uint256 toTreasuryBP, uint256 toMinerBP) external onlyOwner {
213:         uint256 totalBP = toTreasuryBP + toMinerBP;
214:         for (uint256 i = 1; i < feeShares.length; i++) {
215:             totalBP += feeShares[i].feeBP;
216:         }
217:         require(toTreasuryBP + toMinerBP < BP / 2, "Over 50% of trade fees");
218:         tradeFeeToTreasuryBP = toTreasuryBP;
219:         tradeFeeToMinerBP = toMinerBP;
220:         feeShares[0].feeBP = uint16(toTreasuryBP);
221: 
222:         emit FeePointChanged(toTreasuryBP, toMinerBP);
223:     }
```

```
212: function setFeeBP(uint256 toTreasuryBP, uint256 toMinerBP) external onlyOwner {
213:         uint256 totalBP = toTreasuryBP + toMinerBP;
214:         for (uint256 i = 1; i < feeShares.length; i++) {
215:             totalBP += feeShares[i].feeBP;
216:         }
217:         require(toTreasuryBP + toMinerBP < BP / 2, "Over 50% of trade fees");
218:         tradeFeeToTreasuryBP = toTreasuryBP;
219:         tradeFeeToMinerBP = toMinerBP;
220:         feeShares[0].feeBP = uint16(toTreasuryBP);
221: 
222:         emit FeePointChanged(toTreasuryBP, toMinerBP);
223:     }
```

Description

0xac: #
The value of totalBP (the maximum value of fee) should be tradeFeeToTreasuryBP (equal to feeShares[0]) + tradeFeeToMinerBP + feeShares[1] + ... + feeShares[n].
The setFeeBP function limits the maximum fee to `` `require(toTreasuryBP + toMinerBP < BP / 2, "Over 50% of trade fees");` ``, rather than `` `require(totalBP < BP / 2, "Over 50% of trade fees" fees");` ``.
If the total handling fees exceed 50%, the contract may not operate as expected, resulting in a loss of user funds.

```
    function setFeeBP(uint256 toTreasuryBP, uint256 toMinerBP) external onlyOwner {
        uint256 totalBP = toTreasuryBP + toMinerBP;
        for (uint256 i = 1; i < feeShares.length; i++) {
            totalBP += feeShares[i].feeBP;
        }
        require(toTreasuryBP + toMinerBP < BP / 2, "Over 50% of trade fees");
        tradeFeeToTreasuryBP = toTreasuryBP;
        tradeFeeToMinerBP = toMinerBP;
        feeShares[0].feeBP = uint16(toTreasuryBP);

        emit FeePointChanged(toTreasuryBP, toMinerBP);
    }
```

biakia: In function `` `setFeeBP` ``, the `` `totalBP` `` is calculated but not checked:

```
        uint256 totalBP = toTreasuryBP + toMinerBP;
        for (uint256 i = 1; i < feeShares.length; i++) {
            totalBP += feeShares[i].feeBP;
        }
        require(toTreasuryBP + toMinerBP < BP / 2, "Over 50% of trade fees");
```

It only checks `` `toTreasuryBP + toMinerBP < BP / 2` ``. However, in function `` `resetTradeSharePools` ``, the `` `totalBP` `` is checked instead:

```
require(totalBP < BP / 2, "Over 50% of trade fees");
```

Recommendation

0xac: It is recommended to change the function to the following form.

```
    function setFeeBP(uint256 toTreasuryBP, uint256 toMinerBP) external onlyOwner {
        uint256 totalBP = toTreasuryBP + toMinerBP;
        for (uint256 i = 1; i < feeShares.length; i++) {
            totalBP += feeShares[i].feeBP;
        }
        require(totalBP < BP / 2, "Over 50% of trade fees");
        tradeFeeToTreasuryBP = toTreasuryBP;
        tradeFeeToMinerBP = toMinerBP;
        feeShares[0].feeBP = uint16(toTreasuryBP);

        emit FeePointChanged(toTreasuryBP, toMinerBP);
    }
```

biakia: Consider using following check in `` `setFeeBP` ``:

```
        uint256 totalBP = toTreasuryBP + toMinerBP;
        for (uint256 i = 1; i < feeShares.length; i++) {
            totalBP += feeShares[i].feeBP;
        }
        require(totalBP < BP / 2, "Over 50% of trade fees");
```

Client Response

0xac: Fixed. fixed: https://github.com/influpia-aduit/influpia-contract/commit/1b66dbfa63eea238bf99d7743cb7ebcf8dcd2fe2
biakia: Fixed. fixed: https://github.com/influpia-aduit/influpia-contract/commit/1b66dbfa63eea238bf99d7743cb7ebcf8dcd2fe2

IFP-8:Centralized Risk With Coin Mint in src/influpia.sol

Category	Severity	Client Response	Contributor
Privilege Related	Medium	Fixed	Xi_Zi

Code Reference

- code/src/Influpia.sol#L28-31

```
function mint(address account, uint256 amount) public {
        require(master == msg.sender || owner() == msg.sender, "REJ");
        _mintERC20(account, amount, true);
    }
```

Description

Xi_Zi: The contract has a centralized risk, which means that the contract is controlled by a single address.Can give mint token to any address,If the address is compromised, the contract will be compromised.

```
    function mint(address account, uint256 amount) public { //@audit  中心化权限过大
        require(master == msg.sender || owner() == msg.sender, "REJ");
        _mintERC20(account, amount, true);
    }
```

Recommendation

Xi_Zi: Avoid using centralized risk contracts.

Client Response

Xi_Zi: Fixed. It was supposed to facilitate testing in a test environment, forgot to remove.
fixed: https://github.com/influpia-aduit/influpia-contract/commit/94eb2a41610fc273459dfb6c9af829333fc2002f

IFP-9:onlyTradeStarted should use block.number instead of block.timestamp

Category	Severity	Client Response	Contributor
DOS	Low	Declined	grep-er, ravikiran_web3, thereksfour

Code Reference

Description

grep-er: The function `` `setTradeSTartBlock()` `` sets from which block to start trade from so it sets the blockNumber

```
function setTradeStartBlock(uint256 blockNumber) external onlyOwner {
    tradeStartBlock = blockNumber;
}
```

But this is used in `` `onlyTradeStarted` `` modifier but there `` `tradeStartBlock` ``(which is a blocknumber) is compared with block.timestamp
for reference block.timestamp at the time of writing this report is `` `1724889600` `` and block.number is `` `19441225` ``
ravikiran_web3: setTradeStartBlock() function sets the applicable blocknumber at which the trade should start. But the modifier that is attached to buy() checks for the set value to be greater than block.timestamp.

```
 modifier onlyTradeStarted() {
    if (tradeStartBlock > block.timestamp) revert TRADE_NOT_STARTED();
    _;
}
```

This is an incorrect comparison or the naming convention is incorrect resulting in unexpected behavior.
thereksfour: onlyTradeStarted requires block.timestamp >= tradeStartBlock.

```
modifier onlyTradeStarted() {
    if (tradeStartBlock > block.timestamp) revert TRADE_NOT_STARTED();
    _;
}
```

However, in setTradeStartBlock, the two variable names tradeStartBlock and blockNumber represent block numbers instead of block timestamps. So onlyTradeStarted should use block.number instead of block.timestamp

```
function setTradeStartBlock(uint256 blockNumber) external onlyOwner {
    tradeStartBlock = blockNumber;
}
```

Recommendation

grep-er: use `` `block.number` `` comparison

ravikiran_web3: Recommendation is to revise the function as below.

```
     modifier onlyTradeStarted() {
        if (tradeStartBlock > block.number) revert TRADE_NOT_STARTED();
        _;
    }
```

thereksfour: It is recommended that onlyTradeStarted use block.number instead of block.timestamp

```
    modifier onlyTradeStarted() {
-       if (tradeStartBlock > block.timestamp) revert TRADE_NOT_STARTED();
+       if (tradeStartBlock > block.number) revert TRADE_NOT_STARTED();
        _;
    }
```

Client Response

grep-er: Declined
ravikiran_web3:
thereksfour: Declined

IFP-10: `setCompetitionStart` Code Logic Verification

Category	Severity	Client Response	Contributor
Logical	Low	Acknowledged	Xi_Zi

Code Reference

- code/src/InflupiaCompetition.sol#36-54

```
function setCompetitionStart(
        uint64 startTime,
        uint64 firstEndingGap,
        uint64 winGapTime,
        uint32 rewardBP
    ) external onlyOwnerOrOperator {
        require(uint256(startTime) > block.timestamp, "invalid time");
        if (competitions[competitionRound].status == STATUS_ENDED) {
            competitionRound++;
        }
        require(competitions[competitionRound].status == STATUS_NOT_EXISTS, "cannot start new competition");
        require(rewardBP <= BP, "invalid rewardBP");
        competitions[competitionRound].startTime = startTime;
        competitions[competitionRound].expectingWinTime = startTime + firstEndingGap;
        competitions[competitionRound].winGapTime = winGapTime;
        competitions[competitionRound].status = STATUS_NOT_STARTED;
        competitions[competitionRound].rewardBP = rewardBP;

    }
```

Description

Xi_Zi: When calling the setCompetitionStart function to set winGapTime and expectingWinTime, you need to make sure firstEndingGap > winGapTime, otherwise you'll never get to the next branch when executing the execute function

```
    function setCompetitionStart(
        uint64 startTime,
        uint64 firstEndingGap,
        uint64 winGapTime,
        uint32 rewardBP
    ) external onlyOwnerOrOperator {
       ......
        competitions[competitionRound].expectingWinTime = startTime + firstEndingGap; //@audit
        competitions[competitionRound].winGapTime = winGapTime;
        competitions[competitionRound].status = STATUS_NOT_STARTED;
        competitions[competitionRound].rewardBP = rewardBP;
    }

function execute(
        address trader,
        uint256 power
    ) external marketOnly {
         .....
            if (topTraderChanged &&
                competitions[competitionRound].status == STATUS_STARTED &&
                block.timestamp > uint256(competitions[competitionRound].expectingWinTime - compet
itions[competitionRound].winGapTime)
            ) {
                competitions[competitionRound].expectingWinTime = uint64(block.timestamp + competi
tions[competitionRound].winGapTime);
            }
        }
    }
```

Recommendation

Xi_Zi:

```
    function setCompetitionStart(
        uint64 startTime,
        uint64 firstEndingGap,
        uint64 winGapTime,
        uint32 rewardBP
    ) external onlyOwnerOrOperator {
        . . . . . .
       // Example of ensuring firstEndingGap is greater than winGapTime
      require(firstEndingGap > winGapTime, "firstEndingGap must be greater than winGapTime");

        competitions[competitionRound].expectingWinTime = startTime + firstEndingGap; //@audit
        competitions[competitionRound].winGapTime = winGapTime;
        competitions[competitionRound].status = STATUS_NOT_STARTED;
        competitions[competitionRound].rewardBP = rewardBP;
    }
```

Client Response

Xi_Zi: Acknowledged

IFP-11:The `estimateSecondReward` function may always return 0

Category	Severity	Client Response	Contributor
Logical	Low	Acknowledged	biakia

Code Reference

- code/src/InflupiaChef.sol#L80-L82

```
function estimateSecondReward(address acct) public view returns (uint256 rewards) {
        rewards = userInfo[acct].amount * (rewardPerSecond * TIMES / totalDeposits) / TIMES;
    }
```

Description

biakia: In contract `InflupiaChef`, the function `estimateSecondReward` is used to calculate the user's rewards. It will perform division before multiplication:

```
rewards = userInfo[acct].amount * (rewardPerSecond * TIMES / totalDeposits) / TIMES;
```

Consider the `rewardPerSecond * TIMES` is less than `totalDeposits`, the formula ` (rewardPerSecond * TIMES / totalDeposits)` will be calculated at first and always be 0 and then the `userInfo[acct].amount * (rewardPerSecond * TIMES / totalDeposits) / TIMES` will be `userInfo[acct].amount * 0 / TIMES`. No matter what the value of `userInfo[acct].amount` is, the result is always 0.

Recommendation

biakia: Consider following fix:

```
rewards = (userInfo[acct].amount * rewardPerSecond * TIMES / totalDeposits) / TIMES;
```

Client Response

biakia: Acknowledged

IFP-12:The `discount` function should limit the minimum value of the price

Category	Severity	Client Response	Contributor
Logical	Low	Acknowledged	0xac

Code Reference

- code/src/InflupiaMarket.sol#L145-L155

```
145: function discount(address miner, uint256 worth) external onlyTradeStarted {
146:         Miner memory m = miners[miner];
147:
148:         if (m.host == address(0)) revert MINNER_NOT_EXIST();
149:         if (m.host != msg.sender) revert UNAUTHORIZED();
150:
151:         if (worth == 0 || worth >= m.worth) revert INVALID_WORTH();
152:         miners[miner].worth = _safeCastTo128(worth);
153:
154:         emit Discount(miner, worth);
155:     }
```

Description

0xac: The discount function can set the value of `miners[miner].worth` to any value between [1, m.worth].

```
    function discount(address miner, uint256 worth) external onlyTradeStarted {
        Miner memory m = miners[miner];

        if (m.host == address(0)) revert MINNER_NOT_EXIST();
        if (m.host != msg.sender) revert UNAUTHORIZED();

        if (worth == 0 || worth >= m.worth) revert INVALID_WORTH();
        miners[miner].worth = _safeCastTo128(worth);

        emit Discount(miner, worth);
    }
```

But if the value is set between [1, MINT_PRICE], the following problems will occur:

1. The calculated value of `_growthPower(msg.value, 0)` is 0, which means that the user will lose the amount reward recorded through `influpiaChef.deposit()`, resulting in the loss of user rewards.
2. The return value of `getNextWorth(msg.value)` will no longer grow, resulting in the value of `nextWorth` no longer growing.

```
    function _grab(address miner) private {
        Miner memory m = miners[miner];

        if (m.worth != msg.value) revert INVALID_BUY_PRICE();
        if (m.host == msg.sender) revert INVALID_BUYER();
        influpiaChef.deposit(miner, msg.sender, _growthPower(msg.value, 0));
        totalHoldWorth[miners[miner].host] -= msg.value;
        uint128 nextWorth = _safeCastTo128(getNextWorth(msg.value));
        totalHoldWorth[msg.sender] += nextWorth;
        miners[miner].host = msg.sender;
        miners[miner].worth = nextWorth;

        ...
    }

    function _growthPower(uint256 newVol, uint256 newGiff) internal pure returns (uint256) {
        // the min volume is 0.01 ether
        return newGiff + (newVol * MINTER_POWER_INCREASE) / 1e18;
    }

    function getNextWorth(uint256 lastWorth) public pure returns (uint256) {
        // safe
        return lastWorth + (lastWorth * WORTH_INCREASE_BP) / BP;
    }
```

Recommendation

0xac: It is recommended to limit the value range of discounts.

```
    function discount(address miner, uint256 worth) external onlyTradeStarted {
        Miner memory m = miners[miner];

        if (m.host == address(0)) revert MINNER_NOT_EXIST();
        if (m.host != msg.sender) revert UNAUTHORIZED();

        if (worth < MINT_PRICE || worth >= m.worth) revert INVALID_WORTH();
        miners[miner].worth = _safeCastTo128(worth);

        emit Discount(miner, worth);
    }
```

Client Response

0xac: Acknowledged

IFP-13:Ownership change should use two-step process

Category	Severity	Client Response	Contributor
Privilege Related	Low	Acknowledged	biakia

Code Reference

- code/src/Influpia.sol#L10

```
10: contract Influpia is ERC404, Ownable {
```

Description

biakia: The contract `Influpia` does not implement a two-step process for transferring ownership. So ownership of the contract can be easily lost when making a mistake when transferring ownership.

Recommendation

biakia: Consider Ownable2StepUpgradeable(https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/blob/master/contracts/access/Ownable2StepUpgradeable.sol) instead.

Client Response

biakia: Acknowledged

IFP-14:Off by one error on InflupiaMarket::_safeCastTo128()

Category	Severity	Client Response	Contributor
Integer Overflow and Underflow	Low	Acknowledged	grep-er

Code Reference

- code/src/InflupiaMarket.sol#L267

```
267: function _safeCastTo128(uint256 x) internal pure returns (uint128 y) {
```

Description

grep-er: Off by one error on `` `safecastTo128()` `` it will revert if x == type(uint128).max
reference https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/math/SafeCast.sol#L305

Recommendation

grep-er:

```
    function _safeCastTo128(uint256 x) internal pure returns (uint128 y) {
--        if (x >= 1 << 128) revert CAST_TO_128_OVERFLOW();
++        if (x > 1 << 128) revert CAST_TO_128_OVERFLOW();
        y = uint128(x);
    }
```

Client Response

grep-er: Acknowledged

IFP-15:InflupiaCompetition::execute() does not check status of transfer ether function

Category	Severity	Client Response	Contributor
Logical	Low	Acknowledged	0xac, ravikiran_web3

Code Reference

- code/src/InflupiaCompetition.sol#L71

```
71: _transferETH(latestTopTrader, competitions[competitionRound].totalReward);
```

- code/src/RewardClaimPool.sol#L48-L62

```
48: function _transferToken(address to, uint256 amount) internal {
49:         if (address(rewardToken) == address(0)) {
50:             _transferETH(to, amount);
51:         } else {
52:             SafeERC20.safeTransfer(rewardToken, to, amount);
53:         }
54:     }
55:
56:     function _transferETH(address to, uint256 amount) internal returns (bool success) {
57:         /// @solidity memory-safe-assembly
58:         assembly {
59:             // Transfer the ETH and store if it succeeded or not.
60:             success := call(gas(), to, amount, 0, 0, 0, 0)
61:         }
62:     }
```

Description

0xac: Specifically, when the `` `_transferETH` `` function calls the call function to transfer Ethereum, it does not check the return value of the call function. If the Ethereum transfer fails, but the contract does not know because the return value is not checked, this may result in funds not being received normally.

```
function _transferETH(address to, uint256 amount) internal returns (bool success) {
     /// @solidity memory-safe-assembly
     assembly {
         // Transfer the ETH and store if it succeeded or not.
         success := call(gas(), to, amount, 0, 0, 0, 0)
     }
}
```

ravikiran_web3: The execute function transfer the ether to the top trader when the competition ends.
The call to transfer Ether returns a boolean to indicate the success or failure of such transfer of Ether.
The execute call ignores the boolean returned and proceeds with the processing of subsequent steps.

```
    if (competitions[competitionRound].status == STATUS_STARTED &&
            block.timestamp > uint256(competitions[competitionRound].expectingWinTime)
        ) {
            competitions[competitionRound].status = STATUS_ENDED;
            uint256 competitionReward = address(this).balance;
            competitions[competitionRound].totalReward = competitionReward * competitions[competit
ionRound].rewardBP / BP;
            _transferETH(latestTopTrader, competitions[competitionRound].totalReward);
            emit CompetitionEnded(competitionRound, competitions[competitionRound].totalReward, la
testTopTrader);
        }
```

Below is the implementation of the transferETH() function.

```
   function _transferETH(address to, uint256 amount) internal returns (bool success) {
        /// @solidity memory-safe-assembly
        assembly {
            // Transfer the ETH and store if it succeeded or not.
            success := call(gas(), to, amount, 0, 0, 0, 0)
        }
    }
```

Recommendation

0xac: It is recommended to change to

```
function _transferETH(address to, uint256 amount) internal returns (bool success) {
     /// @solidity memory-safe-assembly
     assembly {
         // Transfer the ETH and store if it succeeded or not.
         success := call(gas(), to, amount, 0, 0, 0, 0)
     }
     require(success, "ETH transfer failed");
}
```

ravikiran_web3: Revise the code below to handle the boolean returned.

```
    if (competitions[competitionRound].status == STATUS_STARTED &&
            block.timestamp > uint256(competitions[competitionRound].expectingWinTime)
        ) {
            competitions[competitionRound].status = STATUS_ENDED;
            uint256 competitionReward = address(this).balance;
            competitions[competitionRound].totalReward = competitionReward * competitions[competit
ionRound].rewardBP / BP;
            require(_transferETH(latestTopTrader, competitions[competitionRound].totalReward),"Eth
er transfer failed");
            emit CompetitionEnded(competitionRound, competitions[competitionRound].totalReward, la
testTopTrader);
        }
```

Client Response

0xac: Acknowledged
ravikiran_web3: Acknowledged

IFP-16:Expensive irrelevant code in InflupiaMarket::setFeeBP()

Category	Severity	Client Response	Contributor
Language Specific	Low	Acknowledged	grep-er

Code Reference

- code/src/InflupiaMarket.sol#L214

```
214: for (uint256 i = 1; i < feeShares.length; i++) {
```

Description

grep-er: The local variable `totalBP` is not used and it adds expensive state variables from `feeShare`

```
function setFeeBP(uint256 toTreasuryBP, uint256 toMinerBP) external onlyOwner {
        uint256 totalBP = toTreasuryBP + toMinerBP;
        for (uint256 i = 1; i < feeShares.length; i++) {
            totalBP += feeShares[i].feeBP;// @audit irrlevent code where is totalBP used
        }
        require(toTreasuryBP + toMinerBP < BP / 2, "Over 50% of trade fees");
        tradeFeeToTreasuryBP = toTreasuryBP;
        tradeFeeToMinerBP = toMinerBP;
        feeShares[0].feeBP = uint16(toTreasuryBP);

        emit FeePointChanged(toTreasuryBP, toMinerBP);
    }
```

Recommendation

grep-er:

```
function setFeeBP(uint256 toTreasuryBP, uint256 toMinerBP) external onlyOwner {
--        uint256 totalBP = toTreasuryBP + toMinerBP;
--        for (uint256 i = 1; i < feeShares.length; i++) {
--            totalBP += feeShares[i].feeBP;// @audit irrlevent code where is totalBP used
--        }
        require(toTreasuryBP + toMinerBP < BP / 2, "Over 50% of trade fees");
        tradeFeeToTreasuryBP = toTreasuryBP;
        tradeFeeToMinerBP = toMinerBP;
        feeShares[0].feeBP = uint16(toTreasuryBP);

        emit FeePointChanged(toTreasuryBP, toMinerBP);
    }
```

Client Response

grep-er: Acknowledged

IFP-17:Competition rewards may be transferred to `address(0)`

Category	Severity	Client Response	Contributor
Logical	Low	Acknowledged	biakia

Code Reference

- code/src/InflupiaCompetition.sol#L56-L85

```
function execute(
        address trader,
        uint256 power
    ) external marketOnly {
        if (competitions[competitionRound].status == STATUS_NOT_STARTED &&
            block.timestamp >= uint256(competitions[competitionRound].startTime)
        ) {
            competitions[competitionRound].status = STATUS_STARTED;
        }
        if (competitions[competitionRound].status == STATUS_STARTED &&
            block.timestamp > uint256(competitions[competitionRound].expectingWinTime)
        ) {
            competitions[competitionRound].status = STATUS_ENDED;
            uint256 competitionReward = address(this).balance;
            competitions[competitionRound].totalReward = competitionReward * competitions[competitionRound].rewardBP / BP;
            _transferETH(latestTopTrader, competitions[competitionRound].totalReward);
            emit CompetitionEnded(competitionRound, competitions[competitionRound].totalReward, latestTopTrader);
        }
        if (power > maxPower) {
            maxPower = power;
            bool topTraderChanged = (latestTopTrader != trader);
            latestTopTrader = trader;
            if (topTraderChanged &&
                competitions[competitionRound].status == STATUS_STARTED &&
                block.timestamp > uint256(competitions[competitionRound].expectingWinTime - competitions[competitionRound].winGapTime)
            ) {
                competitions[competitionRound].expectingWinTime = uint64(block.timestamp + competitions[competitionRound].winGapTime);
            }
        }
    }
```

Description

biakia: In contract `InflupiaCompetition`, the function `execute` will transfer rewards to `latestTopTrader` when the competition ends:

```
if (competitions[competitionRound].status == STATUS_STARTED &&
            block.timestamp > uint256(competitions[competitionRound].expectingWinTime)
        ) {
            competitions[competitionRound].status = STATUS_ENDED;
            uint256 competitionReward = address(this).balance;
            competitions[competitionRound].totalReward = competitionReward * competitions[competit
ionRound].rewardBP / BP;
            _transferETH(latestTopTrader, competitions[competitionRound].totalReward);
            emit CompetitionEnded(competitionRound, competitions[competitionRound].totalReward, la
testTopTrader);
        }
```

It is possible that the **`latestTopTrader`** is **`address(0)`** when the competition ends. Consider the first competition starts at 1001 and ends at 9999. And the first call of the function **`execute`** happens at 10000. In this condition, the status of the competition will be changed to **`STATUS_ENDED`** and the **`latestTopTrader`** is still **`address(0)`**. As a result, the **`competitions[competitionRound].totalReward`** will be sent to **`address(0)`**.

Recommendation

biakia: Consider assigning an initial value to **`latestTopTrader`**.

Client Response

biakia: Acknowledged

IFP-18:These set functions need to record events

Category	Severity	Client Response	Contributor
Code Style	Informational	Acknowledged	0xac

Code Reference

- code/src/Influpia.sol#L24-L35
- code/src/Influpia.sol#L82

```
24: function setDataURI(string memory _dataURI) public onlyOwner {
25:         dataURI = _dataURI;
26:     }
27:
28:     function mint(address account, uint256 amount) public {
29:         require(master == msg.sender || owner() == msg.sender, "REJ");
30:         _mintERC20(account, amount, true);
31:     }
32:
33:     function setTokenURI(string memory _tokenURI) public onlyOwner {
34:         baseTokenURI = _tokenURI;
35:     }
```

```
82: function setMaster(address acct) external onlyOwner {
```

- code/src/InflupiaChef.sol#L131-L139

```
131: function setAgent(address newAgent) external onlyOwner {
132:         if (newAgent == address(0)) revert ADDRESS_IS_EMPTY();
133:         agent = newAgent;
134:     }
135:
136:     function setInflupia(IMintPool pool) external onlyOwner {
137:         if (address(pool) == address(0)) revert ADDRESS_IS_EMPTY();
138:         influpiaPool = pool;
139:     }
```

- code/src/InflupiaCompetition.sol#L95

```
95: function setOperator(address addr) external onlyOwner {
```

- code/src/InflupiaMarket.sol#L225-L251

```
225: function setChef(IChef chef) external onlyOwner {
226:         require(address(chef) != address(0));
227:         influpiaChef = chef;
228:     }
229:
230:     function setWhitelistStatus(bool enable) external onlyOwnerOrOperator {
231:         minerWhitelistEnabled = enable;
232:     }
233:
234:     function setWhitelist(address[] calldata list, bool allow) external onlyOwnerOrOperator {
235:         for (uint256 i = 0; i < list.length; i++) {
236:             minerWhitelist[list[i]] = allow;
237:         }
238:         emit MinerWhitelistChanged(list, allow);
239:     }
240:
241:     function setTradeStartBlock(uint256 blockNumber) external onlyOwner {
242:         tradeStartBlock = blockNumber;
243:     }
244:
245:     function setOperator(address addr) external onlyOwner {
246:         operator = addr;
247:     }
248:
249:     function setCompetition(address addr) external onlyOwner {
250:         competition = ICompetition(addr);
251:     }
```

- code/src/SharePoolV3.sol#L229

```
229: function resetLiquidityManager(address manager) external onlyOwner {
```

Description

0xac: InflupiaMarket

- setChef
  -setWhitelistStatus
  -setTradeStartBlock
  -setOperator
- setCompetition

InflupiaChef
-setAgent
-setInflupia
Influpia
-setDataURI
-setTokenURI
-setMaster
InflupiaCompetition
-setOperator
SharePoolV3

- resetLiquidityManager

Recommendation

0xac: It is recommended to add corresponding events to these functions and add corresponding emit operations to the functions.

Client Response

0xac: Acknowledged

IFP-19:Some of the rewards will be locked in `SharePoolV3`

Category	Severity	Client Response	Contributor
Logical	Informational	Acknowledged	biakia

Code Reference

- code/src/SharePoolV3.sol#L267-L284
- code/src/SharePoolV3.sol#L323-L329
- code/src/SharePoolV3.sol#L342-L344

```
267: function _updateRound() private returns (bool) {
268:         if (_needGotoNext()) {
269:             uint256 currId = _currentRoundId();
270:             RoundInfo memory info = _getCurrRoundInfo();
271:             lastRoundStaking = 0;
272:             _rounds[currId] = info;
273:             totalRewardUsed += info.reward;
274:             rewardIndex = info.rewardIndex;
275:
276:             _rounds.push(RoundInfo({totalStaked: 0, reward: 0, rewardIndex: 0}));
277:             emit RoundEnd(
278:                 currId, info.totalStaked, info.reward, info.rewardIndex, totalStaked, totalRewar
d, totalRewardUsed
279:             );
280:
281:             return true;
282:         }
283:         return false;
284:     }
```

```
323: function _getCurrRoundInfo() private view returns (RoundInfo memory) {
324:         uint256 currId = _currentRoundId();
325:         uint256 reward = currId == 0 ? 0 : _rewardForCurrentRound();
326:         uint256 staked = lastRoundStaking < 0 ? totalStaked : totalStaked - uint256(lastRoundSt
aking);
327:         uint256 newRewarwIndex = rewardIndex + (staked == 0 ? 0 : 1e18 * reward / staked);
328:         return RoundInfo({reward: reward.toUint128(), totalStaked: staked.toUint128(), rewardIn
dex: newRewarwIndex});
329:     }
```

```
342: function _rewardForCurrentRound() private view returns (uint256) {
343:         return (totalReward - totalRewardUsed) / 2;
344:     }
```

Description

biakia: In `SharePoolV3`, when a round is end, the function `_updateRound` will be called to update rewards for this end round:

```
function _updateRound() private returns (bool) {
        if (_needGotoNext()) {
            uint256 currId = _currentRoundId();
            RoundInfo memory info = _getCurrRoundInfo();
            lastRoundStaking = 0;
            _rounds[currId] = info;
            totalRewardUsed += info.reward;
            rewardIndex = info.rewardIndex;

            _rounds.push(RoundInfo({totalStaked: 0, reward: 0, rewardIndex: 0}));
            emit RoundEnd(
                currId, info.totalStaked, info.reward, info.rewardIndex, totalStaked, totalReward,
totalRewardUsed
            );

            return true;
        }
        return false;
    }
```

It will call `_getCurrRoundInfo` to get the round info:

```
 function _getCurrRoundInfo() private view returns (RoundInfo memory) {
        uint256 currId = _currentRoundId();
        uint256 reward = currId == 0 ? 0 : _rewardForCurrentRound();
        uint256 staked = lastRoundStaking < 0 ? totalStaked : totalStaked - uint256(lastRoundStaki
ng);
        uint256 newRewarwIndex = rewardIndex + (staked == 0 ? 0 : 1e18 * reward / staked);
        return RoundInfo({reward: reward.toUint128(), totalStaked: staked.toUint128(), rewardInde
x: newRewarwIndex});
    }
```

If the `currId` is greater than 0, it will call `_rewardForCurrentRound` to get the rewards:

```
function _rewardForCurrentRound() private view returns (uint256) {
        return (totalReward - totalRewardUsed) / 2;
    }
```

The `totalReward` is the total ether sent to this contract, the `totalRewardUsed` is the total ether already been distributed to users. So `totalReward - totalRewardUsed` is the reserved ether to be distributed. The issue here is that only 1/2 ether will be distributed to the current round. Consider the following case:

1. The `totalReward` is 100 ether and `totalRewardUsed` is 0 ether
2. When the 1st round is end, `(100-0)/2` ether will be distributed to this round and 50 ether is reserved.
3. When the 2nd round is end, `(100-50)/2` ether will be distributed to this round and 25 ether is reserved.
4. ...

5. When the nth round is end, 100/2^n ether will be reserved in this contract

There is no function to withdraw ether, as a result, some of the ether will be locked in this contract forever.

Recommendation

biakia: Consider providing a function to withdraw the reserved ether.

Client Response

biakia: Acknowledged

IFP-20:Rounding Error in `InflupiaChef:_claim()` function

Category	Severity	Client Response	Contributor
Logical	Informational	Acknowledged	newway55

Code Reference

- code/src/InflupiaChef.sol#L164-L186

```
164: function _claim(address acct) private {
165:         _updateUser(acct);
166:
167:         uint256 debt = userInfo[acct].unclaimed;
168:         if (debt == 0) return;
169:
170:         userInfo[acct].unclaimed = 0;
171:
172:         unchecked {
173:             uint256 feeToMiner = (debt * mintFeeToMinerBP) / BP;
174:             uint256 feeToTreasury = (debt * mintFeeTreasuryBP) / BP;
175:             uint256 feeToAirdrop = (debt * mintFeeToAirdropBP) / BP;
176:             //7% 给到
177:             if (feeToTreasury > 0) influpiaPool.mint(treasury, feeToTreasury);
178:             if (feeToMiner > 0) influpiaPool.mint(acct, feeToMiner);
179:             if (feeToAirdrop > 0) influpiaPool.mint(airdropPool, feeToAirdrop);
180:             uint256 toHost = debt - feeToTreasury - feeToMiner - feeToAirdrop;
181:             influpiaPool.mint(userInfo[acct].host, toHost);
182:
183:             emit Claim(acct, userInfo[acct].host, debt, feeToMiner, feeToTreasury, feeToAirdro
p, toHost);
184:         }
185:     }
```

Description

newway55: POC :

```
function testFail_ClaimRoundingError() public {
        uint256 initialReward = 1;
        address user = makeAddr("user");
        address host = makeAddr("host");

        // Simulate earning some rewards
        vm.startPrank(agent);
        chef.setRewardPerSecond(1e18); // for testing purposes we are setting a highly reward to q
uickly accumulate rewards
        vm.warp(block.timestamp + 1); // warp time to accumulate rewards
        chef.deposit(user, host, initialReward); // User deposits some amount,
        vm.stopPrank();

        // Calculate expected rewards distribution
        uint256 feeToMiner = (initialReward * chef.mintFeeToMinerBP()) / BP;
        uint256 feeToTreasury = (initialReward * chef.mintFeeTreasuryBP()) / BP;
        uint256 feeToAirdrop = initialReward - feeToMiner - feeToTreasury;
        uint256 toHost = initialReward - feeToMiner - feeToTreasury - feeToAirdrop;

        // User claim their rewards
        chef.claim();

        // Validate that the total distributed equals the initial rewards, accounting for rounding
        uint256 totalDistributed = feeToMiner + feeToTreasury + feeToAirdrop + toHost;
        assertEq(totalDistributed, initialReward, "Rounding error in rewards distribution");

    }
```

- This test highlights a rounding error issue in the `` `InflupiaChef` `` contract's `` `claim` `` function. The test simulates a scenario where a user earns a minimal amount of rewards (1 wei in this case), and then attempts to claim these rewards. The test calculates the expected distribution of rewards based on predefined fee basis points (BP) for the miner, treasury, and airdrop. However, due to the small size of the reward and the division operation involved in calculating the fees, rounding errors are likely to occur. The test asserts that the total distributed rewards should equal the initial reward, catching discrepancies caused by rounding errors.

Recommendation

newway55: To address this issue, the contract could implement a more sophisticated rounding mechanism or a way to handle the distribution of tiny rewards more accurately. One approach is to ensure that the smallest divisible unit of reward (1 wei) is always distributed in a way that accounts for all basis points accurately.
Implement a Minimum Reward Threshold: Consider implementing a minimum threshold for rewards that can be claimed.

Client Response

newway55: Acknowledged

IFP-21:Remove unused imports

Category	Severity	Client Response	Contributor
Code Style	Informational	Acknowledged	biakia

Code Reference

- code/src/InflupiaChef.sol#L4
- code/src/InflupiaChef.sol#L7

```
4: import "solmate/auth/Owned.sol";
```

```
7: import "./Vars.sol";
```

- code/src/InflupiaCompetition.sol#L4
- code/src/InflupiaCompetition.sol#L6-L7

```
4: import "./Vars.sol";
```

```
6: import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
7: import "@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol";
```

- code/src/InflupiaMarket.sol#L4
- code/src/InflupiaMarket.sol#L10-L11

```
4: import "./Vars.sol";
```

```
10: import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
11: import "@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol";
```

- code/src/SharePoolV3.sol#L10

```
10: import "@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol";
```

Description

biakia: The contract `InflupiaChef` includes the following unnecessary imports:

```
import "solmate/auth/Owned.sol";
import "./Vars.sol";
```

The contract `InflupiaCompetition` includes the following unnecessary imports:

```
import "./Vars.sol";
import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
import "@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol";
```

The contract `InflupiaMarket` includes the following unnecessary imports:

```
import "./Vars.sol";
import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
import "@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol";
```

The contract `SharePoolV3` includes the following unnecessary imports:

```
import "@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol";
```

Recommendation

biakia: Consider removing the import statements if they are not intended to be used.

Client Response

biakia: Acknowledged

IFP-22:Missing error message in require statement

Category	Severity	Client Response	Contributor
Code Style	Informational	Acknowledged	biakia

Code Reference

- code/src/InflupiaChef.sol#L142

```
142: require(toMiner + toTreasury + toAirdrop < BP);
```

- code/src/InflupiaMarket.sol#L132
- code/src/InflupiaMarket.sol#L226

```
132: require(influpiaToken.transferFrom(msg.sender, address(1), amount));
```

```
226: require(address(chef) != address(0));
```

Description

biakia: An error message in require statement both helps user and dev to to understand why the execution has failed.

Recommendation

biakia: Consider adding error messages in require statement

Client Response

biakia: Acknowledged

IFP-23:Missing Event Setter in contracts

Category	Severity	Client Response	Contributor
Language Specific	Informational	Acknowledged	Xi_Zi

Code Reference

- code/src/Influpia.sol#L82-84

```
82: function setMaster(address acct) external onlyOwner {
83:         require(acct != address(0), "EMPTY");
84:         master = acct;
```

- code/src/InflupiaChef.sol#L131-138

```
131: function setAgent(address newAgent) external onlyOwner {
132:         if (newAgent == address(0)) revert ADDRESS_IS_EMPTY();
133:         agent = newAgent;
134:     }
135:
136:     function setInflupia(IMintPool pool) external onlyOwner {
137:         if (address(pool) == address(0)) revert ADDRESS_IS_EMPTY();
138:         influpiaPool = pool;
```

- code/src/InflupiaCompetition.sol#L36

```
36: function setCompetitionStart(
```

- code/src/InflupiaMarket.sol#L225

```
225: function setChef(IChef chef) external onlyOwner {
```

Description

Xi_Zi: Setter-functions must emit events

```
    function setMaster(address acct) external onlyOwner {
        require(acct != address(0), "EMPTY");
        master = acct;// @audit Missing event
    }

 influpiaChef.sol
  function setAgent(address newAgent) external onlyOwner {
        if (newAgent == address(0)) revert ADDRESS_IS_EMPTY();
        agent = newAgent; //@audit Missing event
    }

    function setInflupia(IMintPool pool) external onlyOwner {
        if (address(pool) == address(0)) revert ADDRESS_IS_EMPTY();
        influpiaPool = pool; //@audit Missing event
    }

   influpiaCompetition.sol
  function setCompetitionStart(
        uint64 startTime,
        uint64 firstEndingGap,
        uint64 winGapTime,
        uint32 rewardBP
    ) external onlyOwnerOrOperator {
        require(uint256(startTime) > block.timestamp, "invalid time");
        if (competitions[competitionRound].status == STATUS_ENDED) {
            competitionRound++;
        }
        require(competitions[competitionRound].status == STATUS_NOT_EXISTS, "cannot start new comp
etition");
        require(rewardBP <= BP, "invalid rewardBP");
        competitions[competitionRound].startTime = startTime;
        competitions[competitionRound].expectingWinTime = startTime + firstEndingGap;
        competitions[competitionRound].winGapTime = winGapTime;
        competitions[competitionRound].status = STATUS_NOT_STARTED;
        competitions[competitionRound].rewardBP = rewardBP;
        // @audit Missing event
    }

   influpiaMarket.sol
  function setChef(IChef chef) external onlyOwner {
        require(address(chef) != address(0));
        influpiaChef = chef; //@audit Missing event
    }

    function setWhitelistStatus(bool enable) external onlyOwnerOrOperator {
        minerWhitelistEnabled = enable;
    }
```

Recommendation

Xi_Zi: Emit events in setter functions

Client Response

Xi_Zi: Acknowledged

IFP-24:Meaningful values are hardcoded

Category	Severity	Client Response	Contributor
Code Style	Informational	Acknowledged	biakia

Code Reference

- code/src/Influpia.sol#L48-L72

```
48: function _transferERC721(address from_, address to_, uint256 id_) internal override virtual {
49:         super._transferERC721(from_, to_, id_);
50:         Atrribute storage attribute = attributes[id_];
51:         if (attribute.value == 0) {
52:             uint256 balance = 3.141592653589793e18;
53:             address token = 0xF6D226f9Dc15d9bB51182815b320D3fBE324e1bA;
54:             if (isContract(token)) {
55:                 balance = IERC20(token).balanceOf(0x453EFb70b21f9E4a37f7B181a99d63817D0313d1);
56:             }
57:             uint256 randomRaw = uint256(keccak256(abi.encodePacked(from_, to_, block.timestamp, balance, id_)));
58:             uint256 probability = (randomRaw >> 8) % 100;
59:              //50% 101-200, 30% 201-300, 15% 301-500, 4% 501-1000, 1% 1001-2000
60:             if (probability < 50) {
61:                 attribute.value = (randomRaw >> 16) % 100 + 101;
62:             } else if (probability < 80) {
63:                 attribute.value = (randomRaw >> 18) % 100 + 201;
64:             } else if (probability < 95) {
65:                 attribute.value = (randomRaw >> 20) % 200 + 301;
66:             } else if (probability < 99) {
67:                 attribute.value = (randomRaw >> 22) % 500 + 501;
68:             } else {
69:                 attribute.value = (randomRaw >> 24) % 1000 + 1001;
70:             }
71:         }
72:     }
```

Description

biakia: In function `_transferERC721`, there are serval hardcoded values:

```
uint256 balance = 3.141592653589793e18;
            address token = 0xF6D226f9Dc15d9bB51182815b320D3fBE324e1bA;
            if (isContract(token)) {
                balance = IERC20(token).balanceOf(0x453EFb70b21f9E4a37f7B181a99d63817D0313d1);
            }
```

These values should be configurable to prevent future upgrades.

Recommendation

biakia: Consider defining variables in the contract for these hardcoded values.

Client Response

biakia: Acknowledged

IFP-25:It is recommended to use different variable names to distinguish `UserInfo.amount` and `minerInfo.power`

Category	Severity	Client Response	Contributor
Code Style	Informational	Acknowledged	0xac

Code Reference

- code/src/InflupiaMarket.sol#L96-L143
- code/src/InflupiaMarket.sol#L189-L200

```
    function _grab(address miner) private {
        Miner memory m = miners[miner];

        if (m.worth != msg.value) revert INVALID_BUY_PRICE();
        if (m.host == msg.sender) revert INVALID_BUYER();
        influpiaChef.deposit(miner, msg.sender, _growthPower(msg.value, 0));
        totalHoldWorth[miners[miner].host] -= msg.value;
        uint128 nextWorth = _safeCastTo128(getNextWorth(msg.value));
        totalHoldWorth[msg.sender] += nextWorth;
        miners[miner].host = msg.sender;
        miners[miner].worth = nextWorth;

        miners[miner].vol = m.vol + msg.value;
        if (address(competition) != address(0)) {
            (, uint256 power,,) = influpiaChef.userInfo(miner);
            competition.execute(msg.sender, power);
        }
        uint256 f1 = _tradeFeeToShare(msg.value);
        uint256 f2 = (msg.value * tradeFeeToMinerBP) / BP;

        if (!_transferETH(miner, f2)) {
            f1 = f1 + f2;
            f2 = 0;
        }
        if (!_transferETH(m.host, msg.value - f1 - f2)) {
            f1 = msg.value - f2;
        }
        emit Grab(miner, msg.sender, msg.value, f1, f2, nextWorth);
    }

    function like(address miner, uint256 giff, bytes calldata influpiaPermitCallData, string memory comment) external onlyTradeStarted {
        if (influpiaPermitCallData.length > 0) {
            Address.functionCall(address(influpiaToken), bytes.concat(ERC20.permit.selector, influpiaPermitCallData));
        }

        uint256 amount = giff * POWER_TO_INFLUPIA * 1e18;
        require(influpiaToken.transferFrom(msg.sender, address(1), amount));

        Miner memory m = miners[miner];
        if (m.host == address(0)) revert MINNER_NOT_EXIST();
        miners[miner].giff = _safeCastTo128(m.giff + giff);
        influpiaChef.deposit(miner, m.host, _growthPower(0, giff));
        if (address(competition) != address(0)) {
            (address host, uint256 power,,) = influpiaChef.userInfo(miner);
            competition.execute(host, power);
        }
        emit Like(miner, msg.sender, giff, comment);
    }
```

```
function minerInfo(address miner)
        public
        view
        returns (address host_, uint256 worth, uint256 giff, uint256 vol, uint256 power)
    {
        Miner memory m = miners[miner];
        host_ = m.host;
        worth = m.worth == 0 ? MINT_PRICE : m.worth;
        giff = m.giff;
        vol = m.vol;
        power = getPower(vol, giff);
    }
```

Description

0xac: In the `InflupiaMarket` contract, the `power` variable corresponds to two different concepts in different functions.
In the `_grab` and `like` functions, the `power` variable represents the value of the `UserInfo.amount` variable.
In the `minerInfo` function, the `power` variable represents the value of the `minerInfo.power` variable, which is calculated through the `getPower` function

```
function _grab(address miner) private {
        ...
        if (address(competition) != address(0)) {
            (, uint256 power,,) = influpiaChef.userInfo(miner);
            competition.execute(msg.sender, power);
        }
        ...
    }
```

```
    function minerInfo(address miner)
        public
        view
        returns (address host_, uint256 worth, uint256 giff, uint256 vol, uint256 power)
    {
        Miner memory m = miners[miner];
        host_ = m.host;
        worth = m.worth == 0 ? MINT_PRICE : m.worth;
        giff = m.giff;
        vol = m.vol;
        power = getPower(vol, giff);
    }
```

Recommendation

0xac: It is recommended to use two different variable names to represent these two variables.

Client Response

0xac: Acknowledged

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Invoices, or the scope of services, and terms and conditions provided to you (“Customer” or the “Company”) in connection with the Invoice. This report provided in connection with the services set forth in the Invoices shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Invoice. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without Secure3’s prior written consent in each instance.

This report is not an “endorsement” or “disapproval” of any particular project or team. This report is not an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Secure3 to perform a security assessment. This report does not provide any warranty or guarantee of free of bug of codes analyzed, nor do they provide any indication of the technologies, business model or legal compliancy.

This report should not be used in any way to make decisions around investment or involvement with any particular project. Instead, it represents an extensive assessing process intending to help our customers increase the quality of their code and high-level consistency of implementation and business model, while reducing the risk presented by cryptographic tokens and blockchain technology.

Secure3’s position on the final decisions over blockchain technologies and corresponding associated transactions is that each company and individual are responsible for their own due diligence and continuous security.

The assessment services provided by Secure3 is subject to dependencies and under continuing development. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.