F Y E O

Security Assessment of the Balanced Java Contracts

Balanced DAO

April 2023
Version 1.0

Presented by:

FYEO Inc.

PO Box 147044
Lakewood CO 80214
United States

Security Level
Strictly Confidential

Table of Contents

List of Figures

List of Tables

EXECUTIVE SUMMARY

Overview

Balanced DAO engaged FYEO Inc. to perform a Security Assessment of the Balanced Java Contracts.

The assessment was conducted remotely by the FYEO Security Team. Testing took place on February 27 - April 03, 2023, and focused on the following objectives:

- To provide the customer with an assessment of their overall security posture and any risks that were discovered within the environment during the engagement.
- To provide a professional opinion on the maturity, adequacy, and efficiency of the security measures that are in place.
- To identify potential issues and include improvement recommendations based on the results of our tests.

This report summarizes the engagement, tests performed, and findings. It also contains detailed descriptions of the discovered vulnerabilities, steps the FYEO Security Team took to identify and validate each issue, as well as any applicable recommendations for remediation.

Key Findings

The following issues were identified during the testing period. They have since been remediated:

- FYEO-BL-01 – Reentrancy vulnerability in stakeICX function
- FYEO-BL-02 – Lack of validation for duration limits in GovernanceImpl
- FYEO-BL-03 – Negative values are not checked in GovernanceImpl
- FYEO-BL-04 – Negative values are not checked in LoansImpl
- FYEO-BL-05 – Token symbol is not check for duplications

Based on our review process, we conclude that the reviewed code implements the documented functionality.

Scope and Rules of Engagement

The FYEO Review Team performed a Security Assessment of the Balanced Java Contracts. The following table documents the targets in scope for the engagement. No additional systems or resources were in scope for this assessment.

The source code was supplied through a public repository at https://github.com/balancednetwork/balanced-java-contracts/tree/main/core-contracts with the commit hash 242955ad6d50230e8ec9fe9eaee92564789c3b91.

A re-review was carried out on commit hash: 3d60070696578c3963cceb0140672c38674f070a

Files included in the code review

```
balanced-java-contracts/
├── core-contracts/
│   ├── BalancedOracle/
│   │   ├── src/
│   │   │   ├── main/
│   │   │   │   └── java/
│   │   │   │       └── network/
│   │   │   │           └── balanced/
│   │   │   │               └── score/
│   │   │   │                   └── core/
│   │   │   │                       └── balancedoracle/
│   │   │   │                           ├── BalancedOracleConstants.java
│   │   │   │                           ├── BalancedOracleImpl.java
│   │   │   │                           ├── EMACalculator.java
│   │   │   │                           └── TWAPCalculator.java
│   ├── DAOfund/
│   │   ├── src/
│   │   │   ├── intTest/
│   │   │   │   └── java/
│   │   │   │       └── network/
│   │   │   │           └── balanced/
│   │   │   │               └── score/
│   │   │   │                   └── core/
│   │   │   │                       └── daofund/
│   │   │   │                           └── DaofundIntegrationTest.java
│   │   │   ├── main/
│   │   │   │   └── java/
│   │   │   │       └── network/
│   │   │   │           └── balanced/
│   │   │   │               └── score/
│   │   │   │                   └── core/
│   │   │   │                       └── daofund/
│   │   │   │                           ├── DAOfundImpl.java
│   │   │   │                           └── POLManager.java
│   ├── Dex/
│   │   ├── src/
│   │   │   ├── main/
```

Files included in the code review

```
│   │   │   │   └── java/
│   │   │   │       └── network/
│   │   │   │           └── balanced/
│   │   │   │               └── score/
│   │   │   │                   └── core/
│   │   │   │                       └── dex/
│   │   │   │                           ├── db/
│   │   │   │                           │   ├── LinkedListDB.java
│   │   │   │                           │   └── NodeDB.java
│   │   │   │                           ├── utils/
│   │   │   │                           │   ├── Check.java
│   │   │   │                           │   ├── Const.java
│   │   │   │                           │   └── LPMetadataDB.java
│   │   │   │                           ├── AbstractDex.java
│   │   │   │                           ├── DexDBVariables.java
│   │   │   │                           └── DexImpl.java
│   │   ├── Governance/
│   │   │   ├── src/
│   │   │   │   ├── main/
│   │   │   │   │   └── java/
│   │   │   │   │       └── network/
│   │   │   │   │           └── balanced/
│   │   │   │   │               └── score/
│   │   │   │   │                   └── core/
│   │   │   │   │                       └── governance/
│   │   │   │   │                           ├── proposal/
│   │   │   │   │                           │   ├── ProposalDB.java
│   │   │   │   │                           │   ├── ProposalManager.java
│   │   │   │   │                           │   └── ProposalStatus.java
│   │   │   │   │                           ├── utils/
│   │   │   │   │                           │   ├── ArbitraryCallManager.java
│   │   │   │   │                           │   ├── ContractManager.java
│   │   │   │   │                           │   ├── EmergencyManager.java
│   │   │   │   │                           │   ├── EventLogger.java
│   │   │   │   │                           │   ├── GovernanceConstants.java
│   │   │   │   │                           │   └── SetupManager.java
│   │   │   │   │                           └── GovernanceImpl.java
│   │   ├── Loans/
│   │   │   ├── src/
│   │   │   │   ├── main/
│   │   │   │   │   └── java/
│   │   │   │   │       └── network/
│   │   │   │   │           └── balanced/
```

Files included in the code review

```
│   │   │   │                   └── score/
│   │   │   │                       └── core/
│   │   │   │                           └── loans/
│   │   │   │                               ├── collateral/
│   │   │   │                               │   └── CollateralDB.java
│   │   │   │                               ├── debt/
│   │   │   │                               │   └── DebtDB.java
│   │   │   │                               ├── linkedlist/
│   │   │   │                               │   ├── LinkedListDB.java
│   │   │   │                               │   └── Node.java
│   │   │   │                               ├── positions/
│   │   │   │                               │   ├── Position.java
│   │   │   │                               │   └── PositionsDB.java
│   │   │   │                               ├── utils/
│   │   │   │                               │   ├── Checks.java
│   │   │   │                               │   ├── IdFactory.java
│   │   │   │                               │   ├── LoansConstants.java
│   │   │   │                               │   ├── PositionBatch.java
│   │   │   │                               │   ├── Standing.java
│   │   │   │                               │   └── TokenUtils.java
│   │   │   │                               ├── LoansImpl.java
│   │   │   │                               └── LoansVariables.java
│   │   ├── Reserve/
│   │   │   ├── src/
│   │   │   │   ├── main/
│   │   │   │   │   └── java/
│   │   │   │   │       └── network/
│   │   │   │   │           └── balanced/
│   │   │   │   │               └── score/
│   │   │   │   │                   └── core/
│   │   │   │   │                       └── reserve/
│   │   │   │   │                           └── ReserveFund.java
│   │   ├── Rewards/
│   │   │   ├── src/
│   │   │   │   ├── main/
│   │   │   │   │   └── java/
│   │   │   │   │       └── network/
│   │   │   │   │           └── balanced/
│   │   │   │   │               └── score/
│   │   │   │   │                   └── core/
│   │   │   │   │                       └── rewards/
│   │   │   │   │                           ├── utils/
│   │   │   │   │                           │   ├── BalanceData.java
```

Files included in the code review

```
│   │   │   │                                   │   └── RewardsConstants.java
│   │   │   │                                   ├── weight/
│   │   │   │                                   │   └── SourceWeightController.java
│   │   │   │                                   ├── DataSourceDB.java
│   │   │   │                                   ├── DataSourceImpl.java
│   │   │   │                                   └── RewardsImpl.java
│   │   ├── Router/
│   │   │   ├── src/
│   │   │   │   ├── main/
│   │   │   │   │   └── java/
│   │   │   │   │       └── network/
│   │   │   │   │           └── balanced/
│   │   │   │   │               └── score/
│   │   │   │   │                   └── core/
│   │   │   │   │                       └── router/
│   │   │   │   │                           └── RouterImpl.java
│   │   ├── StakedLP/
│   │   │   ├── src/
│   │   │   │   ├── main/
│   │   │   │   │   └── java/
│   │   │   │   │       └── network/
│   │   │   │   │           └── balanced/
│   │   │   │   │               └── score/
│   │   │   │   │                   └── core/
│   │   │   │   │                       └── stakedlp/
│   │   │   │   │                           └── StakedLPImpl.java
│   │   └── Staking/
│   │       ├── src/
│   │       │   ├── main/
│   │       │   │   └── java/
│   │       │   │       └── network/
│   │       │   │           └── balanced/
│   │       │   │               └── score/
│   │       │   │                   └── core/
│   │       │   │                       └── staking/
│   │       │   │                           ├── db/
│   │       │   │                           │   ├── Delegation.java
│   │       │   │                           │   ├── DelegationListDB.java
│   │       │   │                           │   ├── LinkedListDB.java
│   │       │   │                           │   └── NodeDB.java
│   │       │   │                           ├── utils/
│   │       │   │                           │   ├── Checks.java
│   │       │   │                           │   ├── Constant.java
```

Files included in the code review

```
│   │   │                                   │   └── UnstakeDetails.java
│   │   │                                   └── StakingImpl.java
├── token-contracts/
│   ├── BalancedDollar/
│   │   ├── src/
│   │   │   ├── main/
│   │   │   │   └── java/
│   │   │   │       └── network/
│   │   │   │           └── balanced/
│   │   │   │               └── score/
│   │   │   │                   └── tokens/
│   │   │   │                       └── balanceddollar/
│   │   │   │                           └── BalancedDollarImpl.java
│   │   └── build.gradle
│   ├── BalancedToken/
│   │   ├── src/
│   │   │   ├── main/
│   │   │   │   └── java/
│   │   │   │       └── network/
│   │   │   │           └── balanced/
│   │   │   │               └── score/
│   │   │   │                   └── tokens/
│   │   │   │                       └── balancedtoken/
│   │   │   │                           ├── BalancedTokenImpl.java
│   │   │   │                           ├── Constants.java
│   │   │   │                           └── Status.java
│   │   └── build.gradle
│   ├── Sicx/
│   │   ├── src/
│   │   │   ├── main/
│   │   │   │   └── java/
│   │   │   │       └── network/
│   │   │   │           └── balanced/
│   │   │   │               └── score/
│   │   │   │                   └── tokens/
│   │   │   │                       └── sicx/
│   │   │   │                           └── SicxImpl.java
│   │   └── build.gradle
│   └── bBaln/
│       ├── src/
│       │   ├── main/
│       │   │   └── java/
│       │   │       └── network/
```

Files included in the code review

```
│       │   │               └── balanced/
│       │   │                   └── score/
│       │   │                       └── tokens/
│       │   │                           ├── db/
│       │   │                           │   ├── LockedBalance.java
│       │   │                           │   └── Point.java
│       │   │                           ├── utils/
│       │   │                           │   └── UnsignedBigInteger.java
│       │   │                           ├── AbstractBoostedBaln.java
│       │   │                           ├── BoostedBalnImpl.java
│       │   │                           └── Constants.java
├── util-contracts/
│   └── Stability/
│       ├── src/
│       │   ├── main/
│       │   │   └── java/
│       │   │       └── network/
│       │   │           └── balanced/
│       │   │               └── score/
│       │   │                   └── util/
│       │   │                       └── stability/
│       │   │                           └── StabilityImpl.java
```

Table 1: Scope

TECHNICAL ANALYSES AND FINDINGS

During the Security Assessment of the Balanced Java Contracts, we discovered:

- 1 finding with HIGH severity rating.
- 3 findings with LOW severity rating.
- 1 finding with INFORMATIONAL severity rating.

The following chart displays the findings by severity.

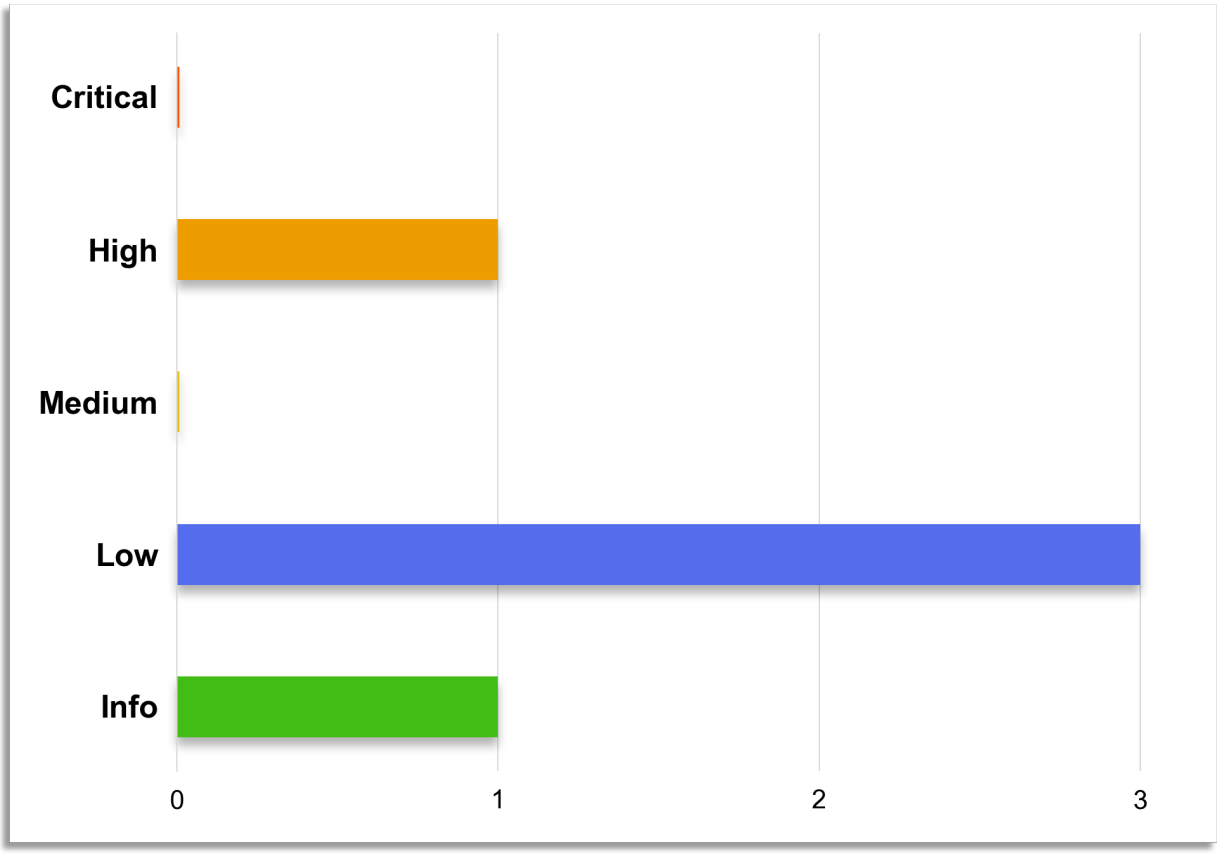

Figure 1: Findings by Severity

Findings

The *Findings* section provides detailed information on each of the findings, including methods of discovery, explanation of severity determination, recommendations, and applicable references.

The following table provides an overview of the findings.

Finding #	Severity	Description
FYEO-BL-01	High	Reentrancy vulnerability in stakeICX function
FYEO-BL-02	Low	Lack of validation for duration limits in GovernanceImpl
FYEO-BL-03	Low	Negative values are not checked in GovernanceImpl
FYEO-BL-04	Low	Negative values are not checked in LoansImpl
FYEO-BL-05	Informational	Token symbol is not check for duplications

Table 2: Findings Overview

Technical Analysis

The source code has been manually validated to the extent that the state of the repository allowed. The validation includes confirming that the code correctly implements the intended functionality.

Technical Findings

General Observations

It is evident that the go code being reviewed has been crafted with care and attention to detail. The development team was quick to respond to our feedback and resolved the issues with great communication and at a rapid pace.

The team's ability to address these issues quickly was a testament to their dedication to producing high-quality code.

While there were some areas for improvement, overall, the code was well-structured and reasonably easy to read.

REENTRANCY VULNERABILITY IN STAKEICX FUNCTION

Finding ID: FYEO-BL-01
Severity: **High**
Status: **Remediated**

Description

The Sicx token `MintTo` function calls `tokenFallback` if the destination address is a contract, so an attacker can supply input `_to` of `statkeICX` with a malicious contract in which its malicious `tokenFallback` function would call `stakeICX` again and reenter the function while the `totalStake` variable is not yet updated. The attacker therefore can manipulate the daily rate so that it can mint a lot more sicx than expected.

Proof of Issue

File name: StakingImpl.java

Line number: 569-582

```
Context.call(sicxAddress.get(), "mintTo", _to, sicxToMint, _data);
        TokenTransfer(_to, sicxToMint, sicxToMint + " sICX minted to " + _to);

        Map<String, BigInteger> userCurrentDelegation =
userDelegationInPercentage.getOrDefault(_to,
                DEFAULT_DELEGATION_LIST).toMap();
        Map<String, BigInteger> prepDelegations =
prepDelegationInIcx.getOrDefault(DEFAULT_DELEGATION_LIST).toMap();
        Map<String, BigInteger> finalDelegation;
        if (!userCurrentDelegation.isEmpty()) {
            finalDelegation = addUserDelegationToPrepDelegation(prepDelegations,
userCurrentDelegation, addedIcx);
        } else {
            finalDelegation = prepDelegations;
        }
        BigInteger newTotalStake =
this.totalStake.getOrDefault(BigInteger.ZERO).add(addedIcx);
        this.totalStake.set(newTotalStake);
```

File name: StakingImpl.java

Line number: 485-501

```
BigInteger dailyReward =
Context.getBalance(Context.getAddress()).subtract(unstakedICX)
                .subtract(msgValue.subtract(icxAdded))
                .subtract(icxToClaim.getOrDefault(BigInteger.ZERO));

        // If there is I-Score generated then update the rate
        if (dailyReward.compareTo(BigInteger.ZERO) > 0) {
            totalLifetimeReward.set(getLifetimeReward().add(dailyReward));
            BigInteger totalStake = getTotalStake();
            BigInteger newTotalStake = totalStake.add(dailyReward);
```

```
            BigInteger newRate;
            if (newTotalStake.equals(BigInteger.ZERO)) {
                newRate = ONE_EXA;
            } else {
                BigInteger totalSupply = (BigInteger) Context.call(sicxAddress.get(),
"totalSupply");
                newRate = newTotalStake.multiply(ONE_EXA).divide(totalSupply);
            }
            rate.set(newRate);
```

Severity and Impact Summary

The attacker is able to mint a lot more sicx than expected; and therefore be able to un-stake more ICX than deposited.

Recommendation

It is recommended to move the mint call to the end of the function.

LACK OF VALIDATION FOR DURATION LIMITS IN GOVERNANCEIMPL

Finding ID: FYEO-BL-02
Severity: Low
Status: Remediated

Description

The `min` input is checked for a positive value, but `max` is not checked if `>= min`.

Proof of Issue

File name: GovernanceImpl.java

Line number: 96-101

```
public void setVoteDurationLimits(BigInteger min, BigInteger max) {
        onlyOwnerOrContract();
        Context.require(min.compareTo(BigInteger.ONE) >= 0, "Minimum vote duration has
to be above 1");
        minVoteDuration.set(min);
        maxVoteDuration.set(max);
    }
```

Severity and Impact Summary

This can lead to incorrect vote duration.

Recommendation

It's recommended to ensure that `max > min`.

NEGATIVE VALUES ARE NOT CHECKED IN GOVERNANCEIMPL

Finding ID: FYEO-BL-03
Severity: Low
Status: Remediated

Description

Both _lockingRatio and _liquidationRatio are checked if > in the loan contract, but _debtCeiling is not checked if positive or not.

Proof of Issue

File name: GovernanceImpl.java

Line number: 460-476

```
public void _addCollateral(Address _token_address, boolean _active, String _peg,
BigInteger _lockingRatio,
                           BigInteger _liquidationRatio, BigInteger _debtCeiling)
{
        Address loansAddress = ContractManager.get("loans");
        Context.call(loansAddress, "addAsset", _token_address, _active, true);

        String symbol = Context.call(String.class, _token_address, "symbol");

        Address balancedOracleAddress = ContractManager.get("balancedOracle");
        Context.call(balancedOracleAddress, "setPeg", symbol, _peg);
        BigInteger price = Context.call(BigInteger.class, balancedOracleAddress,
"getPriceInLoop", symbol);
        Context.require(price.compareTo(BigInteger.ZERO) > 0,
                "Balanced oracle return a invalid icx price for " + symbol + "/" +
_peg);

        Context.call(loansAddress, "setDebtCeiling", symbol, _debtCeiling);
        _setLockingRatio(symbol, _lockingRatio);
        _setLiquidationRatio(symbol, _liquidationRatio);
```

Severity and Impact Summary

This can lead to incorrect calculations of the loans.

Recommendation

It's recommended to ensure that the numbers are positive.

NEGATIVE VALUES ARE NOT CHECKED IN LOANSIMPL

Finding ID: FYEO-BL-04
Severity: Low
Status: Remediated

Description

The `fees`, `points`, and `minimum` are not checked if positive or not.

Proof of Issue

File name: LoansImpl.java

Line number: 790-833

```
@External
    public void setOriginationFee(BigInteger _fee) {
        onlyGovernance();
        originationFee.set(_fee);
    }

    @External
    public void setRedemptionFee(BigInteger _fee) {
        onlyGovernance();
        redemptionFee.set(_fee);
    }

    @External(readonly = true)
    public BigInteger getRedemptionFee() {
        return redemptionFee.get();
    }

    @External
    public void setRedemptionDaoFee(BigInteger _fee) {
        onlyGovernance();
        redemptionDaoFee.set(_fee);
    }

    @External(readonly = true)
    public BigInteger getRedemptionDaoFee() {
        return redemptionDaoFee.getOrDefault(BigInteger.ZERO);
    }

    @External
    public void setRetirementBonus(BigInteger _points) {
        onlyGovernance();
        retirementBonus.set(_points);
    }

    @External
    public void setLiquidationReward(BigInteger _points) {
        onlyGovernance();
        liquidationReward.set(_points);
```

```
    }

    @External
    public void setNewLoanMinimum(BigInteger _minimum) {
        onlyGovernance();
        newLoanMinimum.set(_minimum);
    }
```

Severity and Impact Summary

This can lead to incorrect calculation of the loans.

Recommendation

It's recommended to ensure that the numbers are positive.

TOKEN SYMBOL IS NOT CHECK FOR DUPLICATIONS

Finding ID: FYEO-BL-05
Severity: **Informational**
Status: **Remediated**

Description

The token symbol is not checked if existed when new collateral is added in `addAsset`.

Proof of Issue

File name: LoansImpl.java

Line number: 209-216

```
public void addAsset(Address _token_address, boolean _active, boolean _collateral) {
        onlyGovernance();
        if (_collateral) {
            String symbol = TokenUtils.symbol(_token_address);
            CollateralDB.addCollateral(_token_address, symbol);
            AssetAdded(_token_address, symbol, _collateral);
        }
    }
```

Severity and Impact Summary

Although this function is only called by the governance, it is safer to check for symbol supplications.

Recommendation

Check if the token symbol exists in `CollateralDB`.

OUR PROCESS

METHODOLOGY

FYEO Inc. uses the following high-level methodology when approaching engagements. They are broken up into the following phases.

Figure 2: Methodology Flow

KICKOFF

The project is kicked off as the sales process has concluded. We typically set up a kickoff meeting where project stakeholders are gathered to discuss the project as well as the responsibilities of participants. During this meeting we verify the scope of the engagement and discuss the project activities. It's an opportunity for both sides to ask questions and get to know each other. By the end of the kickoff there is an understanding of the following:

- Designated points of contact
- Communication methods and frequency
- Shared documentation
- Code and/or any other artifacts necessary for project success
- Follow-up meeting schedule, such as a technical walkthrough
- Understanding of timeline and duration

RAMP-UP

Ramp-up consists of the activities necessary to gain proficiency on the project. This can include the steps needed for familiarity with the codebase or technological innovation utilized. This may include, but is not limited to:

- Reviewing previous work in the area including academic papers
- Reviewing programming language constructs for specific languages

- Researching common flaws and recent technological advancements

Review

The review phase is where most of the work on the engagement is completed. This is the phase where we analyze the project for flaws and issues that impact the security posture. Depending on the project this may include an analysis of the architecture, a review of the code, and a specification matching to match the architecture to the implemented code.

In this code audit, we performed the following tasks:

1. Security analysis and architecture review of the original protocol
2. Review of the code written for the project
3. Compliance of the code with the provided technical documentation

The review for this project was performed using manual methods and utilizing the experience of the reviewer. No dynamic testing was performed, only the use of custom-built scripts and tools were used to assist the reviewer during the testing. We discuss our methodology in more detail in the following sections.

Code Safety

We analyzed the provided code, checking for issues related to the following categories:

- General code safety and susceptibility to known issues
- Poor coding practices and unsafe behavior
- Leakage of secrets or other sensitive data through memory mismanagement
- Susceptibility to misuse and system errors
- Error management and logging

This list is general and not comprehensive, meant only to give an understanding of the issues we are looking for.

Technical Specification Matching

We analyzed the provided documentation and checked that the code matches the specification. We checked for things such as:

- Proper implementation of the documented protocol phases

- Proper error handling
- Adherence to the protocol logical description

Reporting

FYEO Inc. delivers a draft report that contains an executive summary, technical details, and observations about the project.

The executive summary contains an overview of the engagement including the number of findings as well as a statement about our general risk assessment of the project. We may conclude that the overall risk is low but depending on what was assessed we may conclude that more scrutiny of the project is needed.

We report security issues identified, as well as informational findings for improvement, categorized by the following labels:

- Critical
- High
- Medium
- Low
- Informational

The technical details are aimed more at developers, describing the issues, the severity ranking and recommendations for mitigation.

As we perform the audit, we may identify issues that aren't security related, but are general best practices and steps that can be taken to lower the attack surface of the project. We will call those out as we encounter them and as time permits.

As an optional step, we can agree on the creation of a public report that can be shared and distributed with a larger audience.

Verify

After the preliminary findings have been delivered, this could be in the form of the approved communication channel or delivery of the draft report, we will verify any fixes within a window of time specified in the project. After the fixes have been verified, we will change the status of the finding in the report from open to remediated.

The output of this phase will be a final report with any mitigated findings noted.

Additional Note

It is important to note that, although we did our best in our analysis, no code audit or assessment is a guarantee of the absence of flaws. Our effort was constrained by resource and time limits along with the scope of the agreement.

While assessing the severity of the findings, we considered the impact, ease of exploitability, and the probability of attack. This is a solid baseline for severity determination.

The Classification of vulnerabilities

Security vulnerabilities and areas for improvement are weighted into one of several categories using, but is not limited to, the criteria listed below:

Critical – vulnerability will lead to a loss of protected assets

- This is a vulnerability that would lead to immediate loss of protected assets
- The complexity to exploit is low
- The probability of exploit is high

High - vulnerability has potential to lead to a loss of protected assets

- All discrepancies found where there is a security claim made in the documentation that cannot be found in the code
- All mismatches from the stated and actual functionality
- Unprotected key material
- Weak encryption of keys
- Badly generated key materials
- Txn signatures not verified
- Spending of funds through logic errors
- Calculation errors overflows and underflows

Medium - vulnerability hampers the uptime of the system or can lead to other problems

- Insecure calls to third party libraries
- Use of untested or nonstandard or non-peer-reviewed crypto functions

- Program crashes, leaves core dumps or writes sensitive data to log files

Low – vulnerability has a security impact but does not directly affect the protected assets

- Overly complex functions
- Unchecked return values from 3rd party libraries that could alter the execution flow

Informational

- General recommendations