FYEO

Security Code Review Aureus Ox

Aureus Ox

June 2024
Version 1.0

Presented by:
FYEO Inc.
PO Box 147044
Lakewood CO 80214
United States

Security Level
Public

Table of Contents

Executive Summary

Overview

Aureus Ox engaged FYEO Inc. to perform a Security Code Review Aureus Ox.

The assessment was conducted remotely by the FYEO Security Team. Testing took place on May 15 - May 22, 2024, 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 have been identified during the testing period. These should be prioritized for remediation to reduce the risk they pose:

- FYEO-AUREUS-OX-01 – No tests have been implemented for the server
- FYEO-AUREUS-OX-02 – Several users can share a user name
- FYEO-AUREUS-OX-03 – Invalid contracts will be scanned
- FYEO-AUREUS-OX-04 – NFTs will be stuck while selling if owner buys another
- FYEO-AUREUS-OX-05 – Some fees have no bounds, not checked to align with user expected values
- FYEO-AUREUS-OX-06 – Typo in userVerification option
- FYEO-AUREUS-OX-07 – Errors are not caught
- FYEO-AUREUS-OX-08 – Missing zero checks
- FYEO-AUREUS-OX-09 – The addAuthenticator function does not return a result

- FYEO-AUREUS-OX-10 – The price defaults to 0
- FYEO-AUREUS-OX-11 – Code clarity
- FYEO-AUREUS-OX-12 – Dependencies not pinned
- FYEO-AUREUS-OX-13 – Use of old Solidity version

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 Code Review Aureus Ox. 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 private repository at https://github.com/Aureus-Ox/aureus-ox-venmo-foundry with the commit hash ae98a352faf819d8b78ac5c47830e7d59b81b091. The remediations were committed with hash 4836731dfd578b9686116f39cfd5cec30669c45d.

Files included in the code review

```
aureus-ox-venmo-foundry/
└── src/
    └── coston/
        ├── implementation/
        │   ├── AddressIdentifier.sol
        │   ├── IdentifierBroker.sol
        │   └── PriceFeed.sol
        └── interface/
            ├── IAddressIdentifier.sol
            ├── IIdentifierBroker.sol
            └── IPriceFeed.sol

api-server/src
├── bigQuery
│   └── jobs
│     └── price.js
├── controllers
│   ├── authentication.js
│   ├── price.js
│   ├── startup.js
```

Files included in the code review

```
│   └── transactions.js
├── covalent
│   ├── transactions.js
│   └── utility.js
├── jwt.js
├── middleware
│   ├── authentication.js
│   └── validation.js
├── models
│   ├── identifier.js
│   ├── price.js
│   ├── transactions.js
│   └── user.js
├── routes
│   ├── authentication.js
│   ├── price.js
│   ├── schema
│   │   ├── login.js
│   │   ├── register.js
│   │   ├── startup.js
│   │   ├── token.js
│   │   └── transactions.js
│   ├── startup.js
│   └── transactions.js
├── secretManager.js
└── spanner
     ├── identifier.js
     ├── instance.js
     └── user.js


iOS
├── AsyncCachingImage
│   ├── AsyncCachingImage.swift
│   ├── ImageCache.swift
│   └── ImageLoader.swift
├── Networking
│   ├── Protocols
│   │   ├── PersistentConnection.swift
│   │   ├── SocketEvent.swift
│   │   └── SocketPayload.swift
│   ├── Rest
│   │   └── Models
```

Files included in the code review

```
│   │       ├── Error.swift
│   │       ├── Request.swift
│   │       └── Requesting.swift
│   ├── Webservice.swift
│   └── Websocket
│     ├── Models
│     │   ├── SocketChannel.swift
│     │   ├── SocketEmission.swift
│     │   └── SubscriptionPayload.swift
│     └── SocketIOService.swift
├── OxenFlow
│   ├── Account
│   │   ├── AccountTabCoordinator.swift
│   │   ├── AccountTabCoordinatorView.swift
│   │   ├── AccountTabModel.swift
│   │   ├── AccountTabView.swift
│   │   └── BackupFactor
│   │       ├── MnemonicPhraseConfirmationCoordinator.swift
│   │       ├── MnemonicPhraseConfirmationModel.swift
│   │       ├── MnemonicPhraseConfirmationView.swift
│   │       └── MnemonicWordView.swift
│   ├── Authentication
│   │   ├── BackupFactor
│   │   │   ├── BackupFactorCoordinator.swift
│   │   │   ├── BackupFactorCoordinatorView.swift
│   │   │   ├── BackupFactorModel.swift
│   │   │   ├── BackupFactorView.swift
│   │   │   ├── BackupPhrase
│   │   │   │   ├── BackupPhraseCoodinator.swift
│   │   │   │   ├── BackupPhraseModel.swift
│   │   │   │   └── BackupPhraseView.swift
│   │   │   └── RecoveryInput
│   │   │       ├── RecoveryInputCoordinator.swift
│   │   │       ├── RecoveryInputCoordinatorView.swift
│   │   │       ├── RecoveryInputModel.swift
│   │   │       └── RecoveryInputView.swift
│   │   ├── Login
│   │   │   ├── LoginCoordinator.swift
│   │   │   ├── LoginModel.swift
│   │   │   └── LoginView.swift
│   │   ├── Register
│   │   │   ├── RegisterCoordinator.swift
│   │   │   ├── RegisterModel.swift
```

Files included in the code review

```
│    │    │    └── RegisterView.swift
│    │    ├── WelcomeCoordinator.swift
│    │    ├── WelcomeCoordinatorView.swift
│    │    ├── WelcomeModel.swift
│    │    └── WelcomeView.swift
│    ├── Colors
│    │    └── Gradient+Extension.swift
│    ├── ContentView.swift
│    ├── Contracts
│    │    ├── AddressIdentifier
│    │    │    ├── ActivateAddress.swift
│    │    │    ├── Approve.swift
│    │    │    ├── BurnIdentifier.swift
│    │    │    ├── IdentifierForAddress.swift
│    │    │    ├── MintIdentifier.swift
│    │    │    ├── OwnerOf.swift
│    │    │    └── TestIdentifier.swift
│    │    └── IdentifierBroker
│    │        ├── BuyIdentifier.swift
│    │        ├── DelistIdentifier.swift
│    │        ├── IdentifierListed.swift
│    │        └── ListIdentifier.swift
│    ├── Dependencies.swift
│    ├── Home
│    │    ├── HomeModel.swift
│    │    └── HomeView.swift
│    ├── Item.swift
│    ├── Marketplace
│    │    ├── Alerts
│    │    │    ├── BurnListIdentifierActionAlertView.swift
│    │    │    └── ClaimIdentifierActionAlertView.swift
│    │    ├── EquippedIdentifier
│    │    │    ├── EquippedIdentifierModel.swift
│    │    │    └── EquippedIdentifierView.swift
│    │    ├── Listed
│    │    │    └── ListedIdentifierThumbnailView.swift
│    │    ├── MarketplaceTabCoordinator.swift
│    │    ├── MarketplaceTabCoordinatorView.swift
│    │    ├── MarketplaceTabModel.swift
│    │    └── MarketplaceTabView.swift
│    ├── Models
│    │    ├── AuthToken.swift
│    │    ├── Requests
```

Files included in the code review

```
│   │   │   ├── Authentication
│   │   │   │   ├── JwksRequest.swift
│   │   │   │   ├── Login
│   │   │   │   │   ├── LoginRequest.swift
│   │   │   │   │   └── LoginVerificationRequest.swift
│   │   │   │   ├── RefreshTokenRequest.swift
│   │   │   │   └── Registration
│   │   │   │       ├── RegistrationRequest.swift
│   │   │   │       └── RegistrationVerificationRequest.swift
│   │   │   ├── Startup
│   │   │   │   └── StartupRequest.swift
│   │   │   └── Transactions
│   │   │       └── UserTransactionsRequest.swift
│   │   ├── Responses
│   │   │   ├── Authentication
│   │   │   │   ├── JwksResponse.swift
│   │   │   │   ├── Login
│   │   │   │   │   └── PublicKeyCredentialRequestOptions.swift
│   │   │   │   ├── RefreshTokenResponse.swift
│   │   │   │   ├── Registration
│   │   │   │   │   └── PublicKeyCredentialCreationOptions.swift
│   │   │   │   └── VerificationResponse.swift
│   │   │   ├── Startup
│   │   │   │   └── StartupResponse.swift
│   │   │   └── Transactions
│   │   │       └── UserTransactionsResponse.swift
│   │   ├── SocketEvents
│   │   │   ├── FtsoEvents.swift
│   │   │   └── IdentifierEvents.swift
│   │   └── User.swift
│   ├── Overview
│   │   ├── Balances
│   │   │   ├── BalancesCardModel.swift
│   │   │   └── BalancesCardView.swift
│   │   ├── OverviewTabModel.swift
│   │   ├── OverviewTabView.swift
│   │   └── Transactions
│   │       ├── TransactionRowModel.swift
│   │       ├── TransactionRowView.swift
│   │       ├── TransactionsModel.swift
│   │       └── TransactionsView.swift
│   ├── OxenFlowApp.swift
│   ├── Payments
```

Files included in the code review

```
│    │    └── Send
│    │        ├── SendSheetModel.swift
│    │        └── SendSheetView.swift
│    ├── ReactiveStore.swift
│    ├── UserAuth.swift
│    ├── Utility
│    │   ├── BundleError.swift
│    │   └── Extensions
│    │       └── View+Extensions.swift
│    ├── ViewComponents
│    │   ├── ButtonComponents
│    │   │   ├── BackButton.swift
│    │   │   └── PrimaryButton.swift
│    │   ├── CustomAlert
│    │   │   ├── CustomAlertModifier.swift
│    │   │   └── CustomAlertView.swift
│    │   ├── Loading
│    │   │   └── Arcs.swift
│    │   ├── PaymentBarView.swift
│    │   ├── SearchBarView.swift
│    │   └── Toast
│    │       ├── ToastMessageModel.swift
│    │       ├── ToastMessageModifier.swift
│    │       └── ToastMessageView.swift
├── OxenFlowTests
│   └── OxenFlowTests.swift
├── OxenFlowUITests
│   ├── OxenFlowUITests.swift
│   └── OxenFlowUITestsLaunchTests.swift
├── SecureStorage
│   └── Keychain
│     └── KeychainAccess.swift
├── Utility
│   ├── EnvironmentVariables.swift
│   ├── Extensions
│   │   ├── ABIFunction+Extensions.swift
│   │   ├── Array+Extensions.swift
│   │   ├── Data+Extensions.swift
│   │   ├── Double+Extensions.swift
│   │   ├── Error+Extensions.swift
│   │   ├── SecKey+Extensions.swift
│   │   └── String+Extensions.swift
│   ├── Jwt.swift
```

Files included in the code review

```
│   ├── Models
│   │   ├── BackupPhrase.swift
│   │   ├── BackupWords.swift
│   │   ├── MnemonicBackup.swift
│   │   ├── RecoveryPhrase.swift
│   │   ├── SigningKey.swift
│   │   └── wordlist.json
│   ├── Protocols
│   │   └── Coordinatable.swift
│   └── Storage
│       └── Protocols
│           └── Storing.swift
└── Web3Auth
    ├── AccountManager+Extension.swift
    ├── AccountManager.swift
    ├── AccountManagerUtils.swift
    ├── Protocols
    │   └── TransactionSigning.swift
    └── Web3Client.swift
```

Table 1: Scope

Technical Analyses and Findings

During the Security Code Review Aureus Ox, we discovered:

- 2 findings with HIGH severity rating.
- 4 findings with MEDIUM severity rating.
- 4 findings with LOW severity rating.
- 3 findings with INFORMATIONAL severity rating.

The following chart displays the findings by severity.

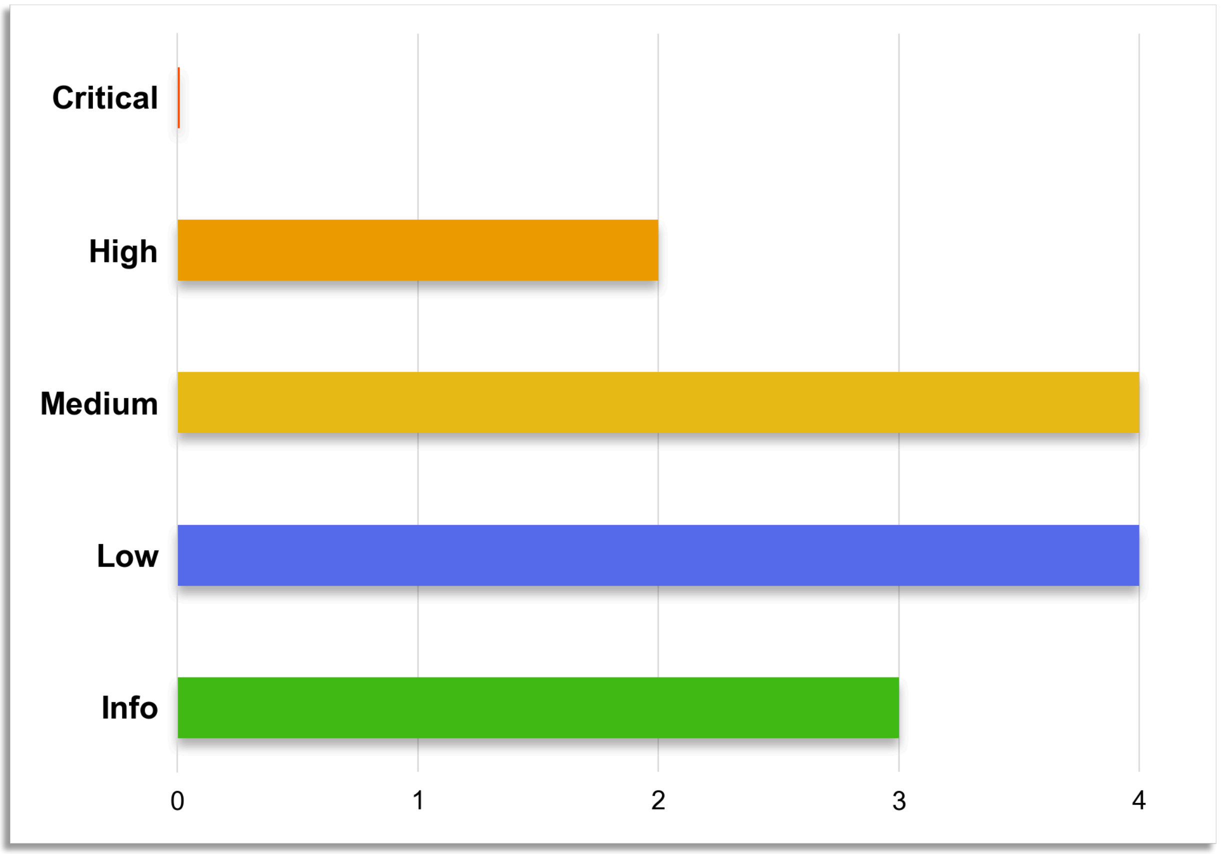

Figure 1: Findings by Severity

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-AUREUS-OX-01	High	No tests have been implemented for the server
FYEO-AUREUS-OX-02	High	Several users can share a user name
FYEO-AUREUS-OX-03	Medium	Invalid contracts will be scanned
FYEO-AUREUS-OX-04	Medium	NFTs will be stuck while selling if owner buys another
FYEO-AUREUS-OX-05	Medium	Some fees have no bounds, not checked to align with user expected values
FYEO-AUREUS-OX-06	Medium	Typo in userVerification option
FYEO-AUREUS-OX-07	Low	Errors are not caught
FYEO-AUREUS-OX-08	Low	Missing zero checks
FYEO-AUREUS-OX-09	Low	The addAuthenticator function does not return a result
FYEO-AUREUS-OX-10	Low	The price defaults to 0
FYEO-AUREUS-OX-11	Informational	Code clarity
FYEO-AUREUS-OX-12	Informational	Dependencies not pinned

FYEO-AUREUS-OX-13	**Informational**	Use of old Solidity version

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.

Conclusion

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

Technical Findings

General Observations

Aureus Ox is an iPhone wallet designed for easy self-custody with an integrated name service. The app is well-structured and leverages the iOS Keychain to securely store data. The use of the SecRandomCopyBytes function guarantees crypto-graphically secure random number generation. The documentation would benefit from some improvement, and the server-side error handling is somewhat lacking. However, the development team is exceptionally responsive, addressing questions and fixing issues immediately. Overall, the app looks promising, and it is clear that security is a priority.

No tests have been implemented for the server

Finding ID: FYEO-AUREUS-OX-01
Severity: **High**
Status: **Remediated**

Description

There are currently no tests implemented for the server.

Proof of Issue

There are no tests.

Severity and Impact Summary

Without proper testing there could be bugs in this server code.

Recommendation

Implement comprehensive tests to verify the integrity of the server implementation.

Several users can share a user name

Finding ID: FYEO-AUREUS-OX-02
Severity: **High**
Status: **Remediated**

Description

It is possible for several users to register the same name. Here is the scenario: Bob registers the username “bob”. User data is inserted, he receives a JWT and will start setting up his authentication. Alice also registers the username “bob”. She still can cause Bob has not finished his setup - which will probably take him several minutes. Bob then completes his registration, the Authenticator data is added and user.registered is set. Alice completes setting up her Authenticator for user “bob” a bit later - it checks out, her Authenticator is also added to user “bob”. Alice and Bob now share access to the account “bob”. For account names with high demand, this could result in a great number of people sharing the account.

Proof of Issue

File name: src/models/user.js
Line number: 7

```
async function registerUser(username) {
    const savedUser = await getUser(username);

    if (!(savedUser)) {
        let userObject = {
            username: username,
            id: uuidv4(),
            registered: false,
            authenticators: []
        }

        // Insert user into big query database
        await saveUser(userObject);
        return userObject;
    } else if (!savedUser.registered) {
        return savedUser;
    }

    return null;
}
```

File name: src/spanner/user.js
Line number: 89

The .registered is only set when an Authenticator is added:

```
async function addAuthenticator(authenticator) {
    const database = instance.database(process.env.DATABASE_ID);

    const registerUserSqlQuery = `UPDATE user
        SET registered = true
        WHERE id = @userId`;
    ...
```

Severity and Impact Summary

Several people can own one account name.

Recommendation

Make sure that only one user can get one account.

Invalid contracts will be scanned

Finding ID: FYEO-AUREUS-OX-03
Severity: **Medium**
Status: **Remediated**

Description

There is a different contract for every chain but the check ignores the actual network and accepts contract addresses from other chains.

Proof of Issue

File name: src/covalent/transactions.js
Line number: 36

```
async function getTransactions(address, network, addressToIdentifier,
addressIdentifierContractAddresses, tokenIdToData) {
    var transactions = []

    try {
        // Eventually we will need to index the chain ourself to do this a better way
        // sql queries and merging the transactions should be able to take care of
        // this use case.
        const chainName = covalentChainName(network);

        for await (const resp of
client.TransactionService.getAllTransactionsForAddress(chainName, address)) {
            ...
            if (addressIdentifierContractAddresses.has(toAddress)) {
```

Severity and Impact Summary

Malicious actors could potentially inject bad data if the wrong contracts are parsed.

Recommendation

Make sure to only allow contracts from the relevant network.

NFTs will be stuck while selling if owner buys another

Finding ID: FYEO-AUREUS-OX-04
Severity: **Medium**
Status: **Remediated**

Description

A user can only own 1 NFT, but they can start selling one and buy another. Now they can't “unlist” the other or swap what's being sold.

Proof of Issue

File name: src/coston/implementation/IdentifierBroker.sol
Line number: 163

```
function delistIdentifier() public nonReentrant {
    require(_sellers[_msgSender()] > 0, "Caller must have item listed for sale");

    uint256 tokenId = _sellers[_msgSender()];

    Identifier storage identifier = _idToIdentifier[tokenId];
    require(block.timestamp - identifier.timeListed >= LISTING_COOLDOWN, "Item cannot
delist until cooldown period ends");

    require(_addressIdentifier.balanceOf(_msgSender()) == 0, "Buyer already has an
identifier equipped");
    require(_msgSender() == identifier.seller, "Something went wrong");

    _addressIdentifier.safeTransfer(address(this), identifier.seller,
identifier.tokenId);

    _sellers[identifier.seller] = 0;
    identifier.listed = false;
    emit IdentifierDelisted(identifier.identifer, identifier.tokenId);
}
```

Severity and Impact Summary

While it isn't much of a security concern, it may be an inconvenience to some users.

Recommendation

Consider if there are some ways to improve this situation.

Some fees have no bounds, not checked to align with user expected values

Finding ID: FYEO-AUREUS-OX-05
Severity: **Medium**
Status: **Remediated**

Description

The fee can be updated to any value and the users have no control as the buy function does not check for discrepancies.

Proof of Issue

File name: src/coston/implementation/IdentifierBroker.sol
Line number: 193

```
function updateCommissionFee(uint256 _percentageFee) public onlyRole(BROKER) {
    COMMISSION_FEE = _percentageFee;
    emit CommissionFeeUpdated(_percentageFee);
}

...

function buyIdentifier(uint256 _tokenId) public payable nonReentrant {
```

These are also unbounded:

```
function updateMintPrice(uint256 _price) public onlyRole(CONTRACT_OWNER) {
    MINT_PRICE = _price * 1e18;
    emit MintPriceUpdated(_price);
}

function updateTokenURIPrice(uint256 _price) public onlyRole(CONTRACT_OWNER) {
    TOKEN_URI_PRICE = _price * 1e18;
    emit TokenURIPriceUpdated(_price);
}

function updateListingCooldown(uint256 _cooldown) public onlyRole(BROKER) {
    LISTING_COOLDOWN = _cooldown;
    emit CooldownPeriodUpdated(_cooldown);
}
```

This could be set to be long after the end of the universe.

Severity and Impact Summary

The user can’t rely on paying a certain fee as the function does not verify what fee the user is expecting to pay. This could be abused by a malicious admin.

Recommendation

Add bounds to the fee and send the fee in the `buyIdentifier` instruction then cancel if that does not match the expectation.

Typo in userVerification option

Finding ID: FYEO-AUREUS-OX-06
Severity: **Medium**
Status: **Remediated**

Description

The `userVerification` setting is set to `requried` rather than `required`.

Proof of Issue

File name: src/controllers/authentication.js
Line number: 219

```
const opts = {
  timeout: 60000,
  // TODO: - Figure out edge case where user doesn't have an authenticator
  // registered. Maybe a cancellation during sign up?
  allowCredentials: user.authenticators.map((authenticator) => ({
    id: authenticator.credentialID,
    type: 'public-key',
    transports: authenticator.transports,
  })),
  /**
   * Wondering why user verification isn't required? See here:
   *
   * https://passkeys.dev/docs/use-cases/bootstrapping/#a-note-about-user-verification
   */
  userVerification: 'requried',
  rpID: this.rpID,
};
```

Severity and Impact Summary

This will likely cause the client to assume this option is set to `preferred`.

Recommendation

Use the correct setting.

Errors are not caught

Finding ID: FYEO-AUREUS-OX-07
Severity: **Low**
Status: **Remediated**

Description

Several functions throw errors which are never caught

Proof of Issue

The functions getUser, registerUser and decryptJWT throw errors. Functions calling these however do not attempt to catch any errors.

Severity and Impact Summary

These errors should be handled to ensure a seamless user experience.

Recommendation

Implement error handling throughout the codebase.

Missing zero checks

Finding ID: FYEO-AUREUS-OX-08
Severity: **Low**
Status: **Remediated**

Description

There are several missing zero checks.

Proof of Issue

File name: src/coston/implementation/PriceFeed.sol
Line number: 39

```
function initialize(IFtsoRegistry ftsoRegistry) public initializer {
    _ftsoRegistry = ftsoRegistry;
```

File name: src/coston/implementation/AddressIdentifier.sol
Line number: 83

```
function initialize(address priceFeed, string memory symbol) public initializer {
```

File name: src/coston/implementation/IdentifierBroker.sol
Line number: 75

```
function initialize(address addressIdentifierContract) public initializer {
```

File name: src/coston/implementation/AddressIdentifier.sol
Line number: 111

```
function addBroker(address broker) public onlyRole(CONTRACT_OWNER) {
```

Severity and Impact Summary

While it isn’t a considerable risk, contracts could be rendered inoperative.

Recommendation

Make sure to check for zero addresses in these places.

The addAuthenticator function does not return a result

Finding ID: FYEO-AUREUS-OX-09
Severity: **Low**
Status: **Remediated**

Description

The `addAuthenticator` function does not indicate whether it executed successfully or not.

Proof of Issue

File name: src/spanner/user.js
Line number: 89

```
async function addAuthenticator(authenticator) { ... }
```

Severity and Impact Summary

In case there is a failure, the function should return an error.

Recommendation

Make sure this function is implemented as intended.

The price defaults to 0

Finding ID: FYEO-AUREUS-OX-10
Severity: **Low**
Status: **Remediated**

Description

The function retrieving the price defaults to 0. It also does not have errors for unexpected inputs such as unknown network ids.

Proof of Issue

File name: src/bigQuery/jobs/price.js
Line number: 4

```
async function fetchLatestPrice(network) {
    var symbol = 'CFLR';

    if (network == 'coston') {
        symbol = 'CFLR'
    } else if (network == 'songbird') {
        symbol = 'SGB'
    }

    ...
    return 0
}
```

Severity and Impact Summary

Parts of the program will consider 0 a legit price.

Recommendation

A clear distinction should be made between an error and an actual price in order to avoid any mistakes in the future.

Code clarity

Finding ID: FYEO-AUREUS-OX-11
Severity: **Informational**
Status: **Remediated**

Description

There are some places where the code can be optimized.

Proof of Issue

File name: src/coston/implementation/AddressIdentifier.sol
Line number: 155

```
require(this.testIdentifier(_identifier), "invalid identifier");
```

The call is local and does not require `this`.

File name: src/coston/implementation/IdentifierBroker.sol
Line number: 148

```
uint256 valueToSend = msg.value - (msg.value - identifier.price);
```

The result will be = `identifier.price` i.e.: `2 - (2 - 1)` is 1.

File name: src/coston/implementation/IdentifierBroker.sol
Line number: 151

```
// Refund the buyer the overpayment
buyer.transfer(msg.value - valueToSend);
```

This function always attempts to send tokens, even if there are none to be send.

Severity and Impact Summary

No security impact.

Recommendation

Keep the code concise to help with the maintainability of the codebase.

Dependencies not pinned

Finding ID: FYEO-AUREUS-OX-12
Severity: **Informational**
Status: **Remediated**

Description

Dependencies are not pinned to specific versions making this service susceptible to supply chain attacks.

Proof of Issue

File name: package.json

```
"dependencies": {
  "@covalenthq/client-sdk": "^1.0.2",
  "@google-cloud/bigquery": "^7.3.0",
  "@google-cloud/secret-manager": "^5.0.1",
  "@google-cloud/spanner": "^7.2.0",
  "@simplewebauthn/server": "^9.0.0",
  "@socket.io/redis-adapter": "^8.2.1",
  "express": "^4.18.2",
  "express-json-validator-middleware": "^3.0.1",
  "helmet": "^7.1.0",
  "hpp": "^0.2.3",
  "jose": "^5.2.0",
  "redis": "^4.6.12",
  "socket.io": "^4.7.4",
  "uuid": "^9.0.1"
}
```

Severity and Impact Summary

This service might upgrade some package to a malicious one.

Recommendation

Pin the dependencies to precise versions and update when deemed safe.

Use of old Solidity version

Finding ID: FYEO-AUREUS-OX-13
Severity: **Informational**
Status: **Remediated**

Description

The program uses Solidity `0.8.13`, which is somewhat old.

Proof of Issue

```
pragma solidity ^0.8.13;
```

Severity and Impact Summary

Newer compiler versions may be able to better optimize the code.

Recommendation

Consider upgrading to a more recent version of the compiler.

Our Process

Methodology

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

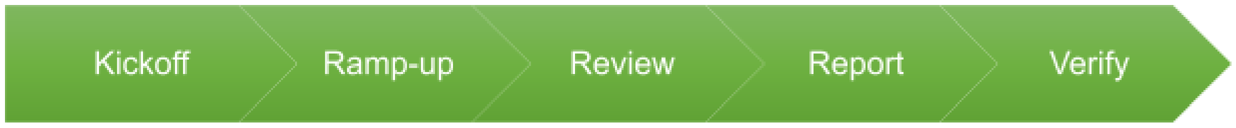

Figure 2: Methodology Flow

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