F Y E O

# Security Code Review of Outcome Protocol

## Outcome

April 2025
Version 1.0

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

Security Level
Public

# Table of Contents

# Executive Summary

## Overview

Outcome engaged FYEO Inc. to perform a Security Code Review of the Outcome Protocol.

The assessment was conducted remotely by the FYEO Security Team. Testing took place on March 10 - March 19, 2025, 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-OUTCOME-01 – Collected fee per sender is overwritten
- FYEO-OUTCOME-02 – Function does not exist
- FYEO-OUTCOME-03 – Check is inverted
- FYEO-OUTCOME-04 – Collateral Token Register & Approval needs improvement
- FYEO-OUTCOME-05 – Distribution is not checked for negative elements
- FYEO-OUTCOME-06 – Overflow concerns
- FYEO-OUTCOME-07 – Batch mint function does not check quantities
- FYEO-OUTCOME-08 – Function is not concise
- FYEO-OUTCOME-09 – Missing quantity and self transfer checks
- FYEO-OUTCOME-10 – No discriminator used in config module

- FYEO-OUTCOME-11 – Possible resource exhaustion in initMarket
- FYEO-OUTCOME-12 – Updating the admin is a one step process
- FYEO-OUTCOME-13 – Arweave address verification can be stricter
- FYEO-OUTCOME-14 – Bound checks
- FYEO-OUTCOME-15 – Duplicate code
- FYEO-OUTCOME-16 – Events are issued without changes occuring
- FYEO-OUTCOME-17 – Mismatched check

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 of Outcome Protocol. 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/puente-ai/outcome-protocol with the commit hash 89d1ab9404a75920798529ac366f048cbf7247ab.

Remediations were submitted with the commit hash ddbbcb75b332814c813a95fabd899e59d10752d2.

| Files included in the code review |
| --- |

```
outcome-protocol/
└── src/
    ├── configuratorModules/
    │   ├── configurator.lua
    │   ├── configuratorNotices.lua
    │   ├── configuratorValidation.lua
    │   └── sharedUtils.lua
    ├── marketFactoryModules/
    │   ├── constants.lua
    │   ├── marketFactory.lua
    │   ├── marketFactoryNotices.lua
    │   ├── marketFactoryValidation.lua
    │   ├── marketProcessCodeV2.lua
    │   ├── sharedUtils.lua
    │   └── sharedValidation.lua
    ├── marketModules/
```

| Files included in the code review |
| --- |
| │ ├── conditionalTokens.lua<br>│ ├── conditionalTokensNotices.lua<br>│ ├── conditionalTokensValidation.lua<br>│ ├── constants.lua<br>│ ├── cpmm.lua<br>│ ├── cpmmHelpers.lua<br>│ ├── cpmmNotices.lua<br>│ ├── cpmmValidation.lua<br>│ ├── json.lua<br>│ ├── market.lua<br>│ ├── marketNotices.lua<br>│ ├── marketValidation.lua<br>│ ├── semiFungibleTokens.lua<br>│ ├── semiFungibleTokensNotices.lua<br>│ ├── semiFungibleTokensValidation.lua<br>│ ├── sharedUtils.lua<br>│ ├── sharedValidation.lua<br>│ ├── sharedValidationOld.lua<br>│ ├── token.lua<br>│ ├── tokenNotices.lua<br>│ ├── tokenValidation.lua<br>│ └── utils.lua<br>├── configurator.lua<br>├── market.lua<br>└── marketFactory.lua |

Table 1: Scope

# Technical Analyses and Findings

During the Security Code Review of the Outcome Protocol, we discovered:

- 1 finding with CRITICAL severity rating.
- 1 finding with HIGH severity rating.
- 4 findings with MEDIUM severity rating.
- 6 findings with LOW severity rating.
- 5 findings with INFORMATIONAL severity rating.

The following chart displays the findings by severity.

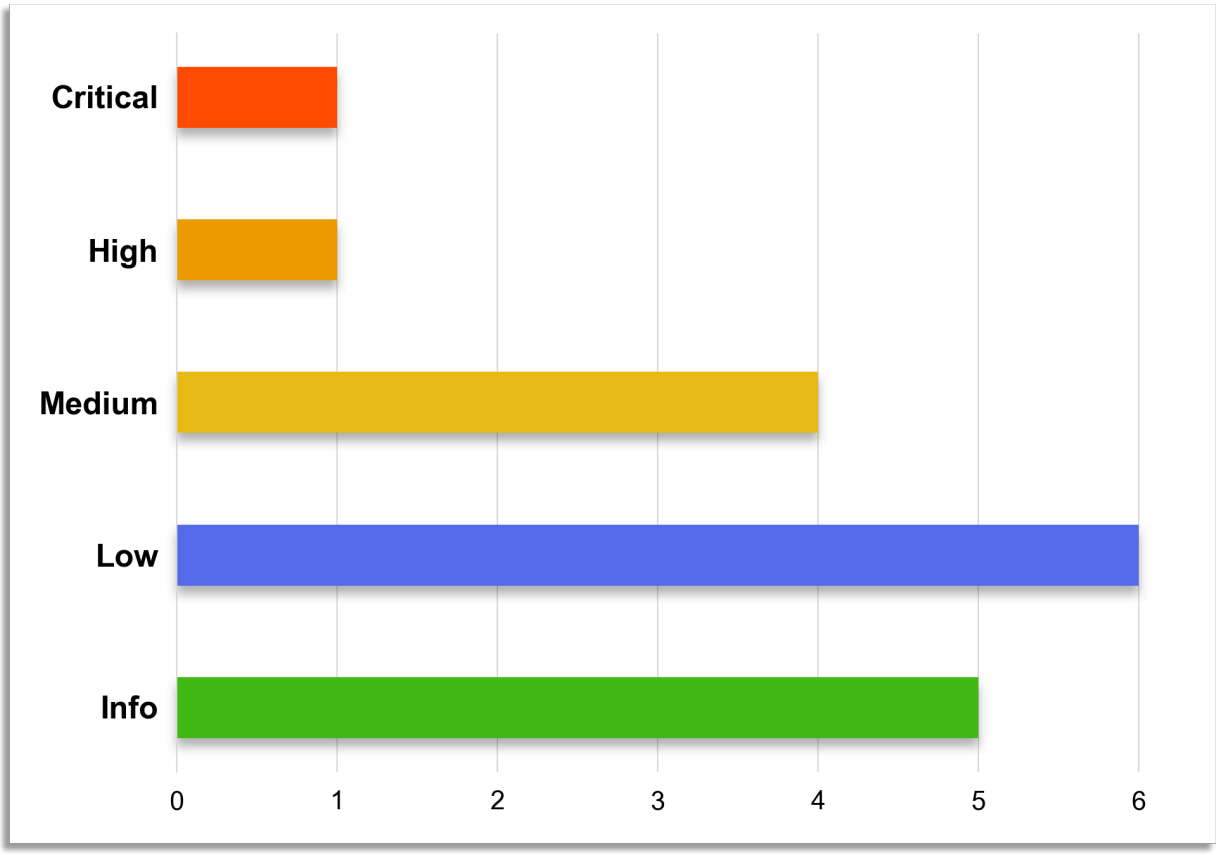


Figure 1: Findings by Severity

## 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-OUTCOME-01 | Critical | Collected fee per sender is overwritten |
| FYEO-OUTCOME-02 | High | Function does not exist |
| FYEO-OUTCOME-03 | Medium | Check is inverted |
| FYEO-OUTCOME-04 | Medium | Collateral Token Register & Approval needs improvement |
| FYEO-OUTCOME-05 | Medium | Distribution is not checked for negative elements |
| FYEO-OUTCOME-06 | Medium | Overflow concerns |
| FYEO-OUTCOME-07 | Low | Batch mint function does not check quantities |
| FYEO-OUTCOME-08 | Low | Function is not concise |
| FYEO-OUTCOME-09 | Low | Missing quantity and self transfer checks |
| FYEO-OUTCOME-10 | Low | No discriminator used in config module |
| FYEO-OUTCOME-11 | Low | Possible resource exhaustion in initMarket |
| FYEO-OUTCOME-12 | Low | Updating the admin is a one step process |
| FYEO-OUTCOME-13 | Informational | Arweave address verification can be stricter |
| FYEO-OUTCOME-14 | Informational | Bound checks |
| FYEO-OUTCOME-15 | Informational | Duplicate code |
| FYEO-OUTCOME-16 | Informational | Events are issued without changes occuring |
| FYEO-OUTCOME-17 | Informational | Mismatched check |

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

The Outcome Protocol is a decentralized automated market maker and prediction market protocol that allows for permissionless market creation and autonomous, oracle-free resolution. The protocol operates on Arweave and AO, ensuring all market data, including outcomes and transactions, is permanently stored on the blockchain.

The protocol enables users to create, fund, trade within, and resolve prediction markets without the need for centralized intermediaries. Market creation involves specifying key parameters such as the collateral token, resolution agent, data index, outcome count, market question, rules, and fee structure. The system is designed to allow liquidity providers to fund markets and set initial probability distributions, while traders can buy and sell outcome positions based on their expectations.

Market resolution is handled autonomously via designated resolution agents, with support for single-signer and zero-resolution (perpetual) markets. The architecture follows a transient-style model where each market operates as an independent process. Core contracts and processes are ownerless, with updates governed through token-based mechanisms.

The protocol emphasizes transparency and efficiency, allowing users to track market probabilities, trades, and resolutions through detailed logs and queries.

During the review of Outcome Protocol's codebase, it was noted that the code base is well structured and that the team has done a high-quality implementation. During the audit the team was very responsive and helped address any questions raised.

## Collected fee per sender is overwritten

Finding ID: FYEO-OUTCOME-01
Severity: **Critical**
Status: **Remediated**

### Description

The `withdrawnFees[sender]` function is overwritten when fees are collected. This should be accumulated instead.

### Proof of Issue

**File name:** src/marketModules/cpmm.lua
**Line number:** 383

```
function CPMMMethods:withdrawFees(sender, onBehalfOf, cast, sendInterim, detached, msg)
  local feeAmount = self:feesWithdrawableBy(sender)
  if bint.__lt(0, bint(feeAmount)) then
    self.withdrawnFees[sender] = feeAmount
    self.totalWithdrawnFees = tostring(bint.__add(bint(self.totalWithdrawnFees),
bint(feeAmount)))
    ao.send({
      Action = 'Transfer',
      Target = self.tokens.collateralToken,
      Recipient = onBehalfOf,
      Quantity = feeAmount,
      ---@diagnostic disable-next-line: assign-type-mismatch
      Cast = not sendInterim and "true" or nil
    })
  end
  if not cast then return self.withdrawFeesNotice(feeAmount, onBehalfOf, detached, msg)
end
end

function CPMMMethods:feesWithdrawableBy(account)
  local balance = self.token.balances[account] or '0'
  local rawAmount = '0'
  if bint(self.token.totalSupply) > 0 then
    rawAmount = string.format('%.0f', (bint.__div(bint.__mul(bint(self:collectedFees()),
bint(balance)), self.token.totalSupply)))
  end
  return tostring(bint.max(bint(bint.__sub(bint(rawAmount),
bint(self.withdrawnFees[account] or '0'))), 0))
end
```

The amount that can be withdrawn is determined by the amount of tokens held (=rawAmount) and the amount already withdrawn `rawAmount - withdrawnFees[account]`. Therefore this should hold a total, not the last withdrawn amount. Otherwise if 200 tokens can be withdrawn, and the user withdraws 100, they can keep withdrawing 100 tokens again and again.

Lastly, the code in `_beforeTokenTransfer` may not work correctly in all instances:

```
if from ~= nil and to ~= nil and from ~= ao.id then
  self.withdrawnFees[from] = tostring(bint.__sub(bint(self.withdrawnFees[from] or '0'),
bint(withdrawnFeesTransfer)))
  self.withdrawnFees[to] = tostring(bint.__add(bint(self.withdrawnFees[to] or '0'),
bint(withdrawnFeesTransfer)))
end
```

This function calculates the withdrawable fees as a share of the transfer amount. This might become negative. Which enables the user to later claim fees. This might be intentional as a way to credit the user while preventing the receiver from claiming. Make sure this intent is clearly documented and matches implementation requirements.

### Severity and Impact Summary

The user can withdraw as many tokens as they please.

### Recommendation

Accumulate the `withdrawnFees` instead of overwriting them.

## Function does not exist

Finding ID: FYEO-OUTCOME-02
Severity: High
Status: Remediated

### Description

The `Update-Market-Process-Code` handler has a call to `marketFactoryValidation.updateMarketProcessCode`. This function does not exist.

### Proof of Issue

**File name:** src/marketFactory.lua
**Line number:** 373

```
local success, err =
marketFactoryValidation.updateMarketProcessCode(MarketFactory.configurator, msg)
```

### Severity and Impact Summary

This code will not work. Therefore the intended functionality is not available.

### Recommendation

Implement the required functionality.

## Check is inverted

Finding ID: FYEO-OUTCOME-03
Severity: Medium
Status: Remediated

### Description

The `ConditionalTokensValidation.mergePositions` and `ConditionalTokensValidation.redeemPositions` functions make sure that `onBehalfOf` is set, then validate the address with `if not onBehalfOf`.

### Proof of Issue

**File name:** src/marketModules/conditionalTokensValidation.lua
**Line number:** 65, 100

```
if not onBehalfOf then
  success, err = sharedValidation.validateAddress(onBehalfOf, 'onBehalfOf')
  if not success then return false, err end
end
```

### Severity and Impact Summary

This means the validation will never be done.

### Recommendation

Remove the `not` to actually run the check.

# Collateral Token Register & Approval needs improvement

Finding ID: FYEO-OUTCOME-04
Severity: Medium
Status: Remediated

## Description

The `Approve-Collateral-Token` and `Register-Collateral-Token` handlers overlap. Tokens can be registered as approved. And the approve function can un-approve, which is not implied by its name. The same for the register function as it does not concern itself with existing data and just overwrites. This needs better separation and needs to consider if the data is already there. Events should only be emitted if things have changed as currently setting an approved token to approved will issue a notification even though nothing changed - same vice-versa.

## Proof of Issue

**File name:** src/marketFactory.lua
**Line number:** 514

```
-- If validation passes, approve collateralToken.
local approved = string.lower(msg.Tags.Approved) == "true"

function marketFactoryValidation.registerCollateralToken(configurator, msg)
  -- no check if collateral exists
```

## Severity and Impact Summary

The code, as implemented, makes it easy to make mistakes that could overwrite existing data or it could be used to disable approved collateral.

## Recommendation

These functions should enforce clearer separation of concerns and consider if data already exists.

## Distribution is not checked for negative elements

Finding ID: FYEO-OUTCOME-05
Severity: Medium
Status: Remediated

### Description

The initial call to `addFunding` has to include a `distribution`. The items in the `distribution` are however not verified to be 0 or positive numbers.

### Proof of Issue

**File name:** src/marketModules/cpmmValidation.lua
**Line number:** 40

```
-- Validate distribution content
local distributionSum = 0
for i = 1, #distribution do
  if type(distribution[i]) ~= "number" then
    return false, "Distribution item must be a number"
  end
  distributionSum = distributionSum + distribution[i]
end
```

### Severity and Impact Summary

Negative elements will result in incorrect calculations.

### Recommendation

Make sure the distribution data is valid.

## Overflow concerns

Finding ID: FYEO-OUTCOME-06
Severity: Medium
Status: Remediated

### Description

The `bint` module does not check for overflows but instead wraps around. This could lead to overflow and underflow issues in the code base.

### Proof of Issue

**File name:** src/marketModules/token.lua
**Line number:** 90

```
self.totalSupply = tostring(bint.__add(bint(self.totalSupply), bint(quantity)))
```

**File name:** src/marketFactoryModules/marketFactoryValidation.lua
**Line number:** 104

```
local totalFee = bint.__add(bint(msg.Tags.CreatorFee), bint(protocolFee))
```

**File name:** src/marketModules/semiFungibleTokens.lua
**Line number:** 86

```
self.balancesById[id][to] = tostring(bint.__add(self.balancesById[id][to],
bint(quantity)))
self.totalSupplyById[id] = tostring(bint.__add(self.totalSupplyById[id],
bint(quantity)))

...

self.balancesById[ ids[i] ][to] = tostring(bint.__add(self.balancesById[ ids[i] ][to],
quantities[i]))
self.totalSupplyById[ ids[i] ] = tostring(bint.__add(self.totalSupplyById[ ids[i] ],
quantities[i]))
```

**File name:** src/marketModules/cpmm.lua
**Line number:** 307, 370, 413

```
local feeAmount = tostring(bint.ceil(bint.__div(bint.__mul(investmentAmount,
self.lpFee), 1e4)))

...

rawAmount = string.format('%.0f', (bint.__div(bint.__mul(bint(self:collectedFees()),
bint(balance)), self.token.totalSupply)))

...

local withdrawnFeesTransfer = totalSupply == '0' and amount or
tostring(bint(bint.__div(bint.__mul(bint(self:collectedFees()), bint(amount)),
totalSupply)))
```

### Severity and Impact Summary

Overflows could lead to critical issues if not properly prevented. Tokens could potentially be minted and overflow the total supply. Adding two fees as `bint` could result in an overflow. This in turn could affect all other protocol calculations.

### Recommendation

Wrap the bint module with checked math operations to guarantee math operations trigger errors should they overflow.

# Batch mint function does not check quantities

Finding ID: FYEO-OUTCOME-07
Severity: Low
Status: Remediated

## Description

The `emiFungibleTokensMethods:batchMint` function is missing the <= 0 check for quantities.

## Proof of Issue

**File name:** src/marketModules/semiFungibleTokens.lua
**Line number:** 103

```
function SemiFungibleTokensMethods:batchMint(to, ids, quantities, cast, detached, msg)
  assert(#ids == #quantities, 'Ids and quantities must have the same lengths')
  -- mint tokens
  for i = 1, #ids do
    -- @dev spacing to resolve text to code eval issue
    if not self.balancesById[ ids[i] ] then self.balancesById[ ids[i] ] = {} end
    if not self.balancesById[ ids[i] ][to] then self.balancesById[ ids[i] ][to] = "0"
end
    if not self.totalSupplyById[ ids[i] ] then self.totalSupplyById[ ids[i] ] = "0" end
    self.balancesById[ ids[i] ][to] = tostring(bint.__add(self.balancesById[ ids[i]
][to], quantities[i]))
    self.totalSupplyById[ ids[i] ] = tostring(bint.__add(self.totalSupplyById[ ids[i] ],
quantities[i]))
  end
```

## Severity and Impact Summary

This has been checked in validators before, but since this is an essential test and actually implemented for the `mint` function, it should still be checked here as well. This will make sure the code won't break if a new handler forgets to check.

## Recommendation

Make sure all quantities are greater than 0.

## Function is not concise

Finding ID: FYEO-OUTCOME-08
Severity: Low
Status: Remediated

### Description

The `SemiFungibleTokensMethods:getBalance` function appears to be a bit un-intuitive as it is not clear whose balance is returned with various cases being supported.

### Proof of Issue

**File name:** src/marketModules/semiFungibleTokens.lua
**Line number:** 233

```
function SemiFungibleTokensMethods:getBalance(sender, recipient, id)
  local bal = '0'
  -- If ID is found then continue
  if self.balancesById[id] then
    -- If recipient is not provided, return the senders balance
    if (recipient and self.balancesById[id][recipient]) then
      bal = self.balancesById[id][recipient]
    elseif self.balancesById[id][sender] then
      bal = self.balancesById[id][sender]
    end
  end
  -- return balance
  return bal
end
```

### Severity and Impact Summary

This function lacks clarity about whose balance is being returned.

### Recommendation

Consider splitting this function into more meaningful individual functions.

## Missing quantity and self transfer checks

Finding ID: FYEO-OUTCOME-09
Severity: **Low**
Status: **Remediated**

### Description

The `SemiFungibleTokensMethods:transferSingle`, `SemiFungibleTokensMethods:transferBatch` and `TokenMethods:transfer` functions do not check the quantity and allow transfer from self to self. This would correctly do nothing, but is probably better to check against. If negative amounts were possible this could enable users to take money from others.

### Proof of Issue

**File name:** src/marketModules/token.lua
**Line number:** 121

```
function TokenMethods:transfer(from, recipient, quantity, cast, detached, msg)
  if not self.balances[from] then self.balances[from] = "0" end
  if not self.balances[recipient] then self.balances[recipient] = "0" end

  local qty = bint(quantity)
```

**File name:** src/marketModules/semiFungibleTokens.lua
**Line number:** 171, 200

```
function SemiFungibleTokensMethods:transferSingle(from, recipient, id, quantity, cast,
detached, msg)
  if not self.balancesById[id] then self.balancesById[id] = {} end
  if not self.balancesById[id][from] then self.balancesById[id][from] = "0" end
  if not self.balancesById[id][recipient] then self.balancesById[id][recipient] = "0"
end

  local qty = bint(quantity)

...

function SemiFungibleTokensMethods:transferBatch(from, recipient, ids, quantities, cast,
detached, msg)
  local ids_ = {}
  local quantities_ = {}

  for i = 1, #ids do
    if not self.balancesById[ ids[i] ] then self.balancesById[ ids[i] ] = {} end
    if not self.balancesById[ ids[i] ][from] then self.balancesById[ ids[i] ][from] =
"0" end
    if not self.balancesById[ ids[i] ][recipient] then self.balancesById[ ids[i]
][recipient] = "0" end

    local qty = bint(quantities[i])
```

### Severity and Impact Summary

The amount checks are handled in validator functions. They are however critical checks that should not rely on external verification.

### Recommendation

Add checks for the quantity and make sure that receiver != sender.

## No discriminator used in config module

Finding ID: FYEO-OUTCOME-10
Severity: **Low**
Status: **Remediated**

### Description

All hashes are stored using `self.staged[hash]`, which means there isn't a way to discern what type of update this hash represents. The use of a discriminator could be useful like `self.staged['configurator' .. hash]` to avoid any collisions between different types.

### Proof of Issue

**File name:** src/configuratorModules/configurator.lua
**Line number:** many instances

```
self.staged[hash] = os.time()
```

### Severity and Impact Summary

For example, if there were another type of low privileged address to update and instead it was interpreted as the configurator address, this could cause severe problems.

### Recommendation

Make sure that the type of update that each hash refers to is encoded in storage.

## Possible resource exhaustion in initMarket

Finding ID: FYEO-OUTCOME-11
Severity: **Low**
Status: **Remediated**

### Description

The `MarketFactoryMethods:initMarket` function has a for loop that calls external functions. This could potentially become large enough to a point where it calls too much, uses too much gas or gets stuck for complexity reasons.

### Proof of Issue

**File name:** src/marketFactoryModules/marketFactory.lua
**Line number:** 433

```
for i = 1, #processIds do
  local processId = processIds[i]
  ao.send({
    Target = processId,
    Action = "Eval",
    Data = self.marketProcessCode,
  })

  ...
```

### Severity and Impact Summary

If resources are exhausted due the size of the work done in the loop, this would become a DoS situation.

### Recommendation

Consider limiting this to X iterations.

## Updating the admin is a one step process

Finding ID: FYEO-OUTCOME-12
Severity: Low
Status: Remediated

### Description

The update of the Configurator is a one step process. A mistake here could lead to a loss of access since the new Configurator does not have to sign. Consider doing this in two steps instead by first proposing a new Configurator and then having the new Configurator take ownership by sending a transaction. This applies to the Market and CPMM.

### Proof of Issue

**File name:** src/marketFactoryModules/marketFactory.lua
**Line number:** 566

```
function MarketFactoryMethods:updateConfigurator(configurator, msg)
  self.configurator = configurator
  return self.updateConfiguratorNotice(configurator, msg)
end
```

**File name:** src/marketModules/cpmm.lua
**Line number:** 445

```
function CPMMMethods:updateConfigurator(configurator, msg)
  self.configurator = configurator
  return self.updateConfiguratorNotice(configurator, msg)
end
```

### Severity and Impact Summary

A mistake may lead to a loss of access.

### Recommendation

Making this a two step process could prevent some mistakes.

## Arweave address verification can be stricter

Finding ID: FYEO-OUTCOME-13
Severity: **Informational**
Status: **Remediated**

### Description

The `isValidArweaveAddress` function could be stricter, as addresses are base64 (See references). So this pattern includes characters that are not valid in the address.

### Proof of Issue

**File name:** src/*/sharedUtils.lua (several occurrences)
**Line number:** 80

```
function sharedUtils.isValidArweaveAddress(address)
    return type(address) == "string" and #address == 43 and string.match(address,
"^[%w-_]+$") ~= nil
end
```

### Severity and Impact Summary

This code may accept invalid addresses.

### Recommendation

Make sure this check matches the set of allowed characters as documented by Arweave.

### References

https://cookbook.arweave.dev/concepts/keyfiles-and-wallets.html#wallet-addresses

## Bound checks

Finding ID: FYEO-OUTCOME-14
Severity: **Informational**
Status: **Remediated**

### Description

Consider checking for upper bounds for settings such as `msg.Tags.StartTime`, `msg.Tags.EndTime`, `msg.Tags.LpFee`, `msg.Tags.MaximumTakeFee`. It may be a mistake if the start time is 1000 years in the future.

### Proof of Issue

**File name:** src/marketFactoryModules/marketFactoryValidation.lua
**Line number:** 51, 112

```
if msg.Tags.StartTime then
  success, err = sharedValidation.validatePositiveInteger(msg.Tags.StartTime,
"StartTime")
  if not success then return false, err end
end

if msg.Tags.EndTime then
  success, err = sharedValidation.validatePositiveInteger(msg.Tags.EndTime, "EndTime")
  if not success then return false, err end
end

...

if msg.Tags.StartTime then
  success, err = sharedValidation.validatePositiveInteger(msg.Tags.StartTime,
"StartTime")
  if not success then return false, err end
end

if msg.Tags.EndTime then
  success, err = sharedValidation.validatePositiveInteger(msg.Tags.EndTime, "EndTime")
  if not success then return false, err end
end

...

function marketFactoryValidation.updateLpFee(configurator, msg)
  if msg.From ~= configurator then
    return false, "Sender must be configurator!"
  end
  return sharedValidation.validatePositiveIntegerOrZero(msg.Tags.LpFee, "LpFee")
end

...

function marketFactoryValidation.updateMaximumTakeFee(configurator, msg)
  if msg.From ~= configurator then
```

```
    return false, "Sender must be configurator!"
  end
  return sharedValidation.validatePositiveIntegerOrZero(msg.Tags.MaximumTakeFee,
"MaximumTakeFee")
end
```

**File name:** src/configuratorModules/configuratorValidation.lua
**Line number:** 57

```
function ConfiguratorValidation.updateDelay(msg)
  if msg.From ~= Configurator.admin then
    return false, 'Sender must be admin!'
  end
  if not msg.Tags.UpdateDelay then
    return false, 'UpdateDelay is required!'
  end

  local delay = tonumber(msg.Tags.UpdateDelay)
  if not delay then
    return false, 'UpdateDelay must be a valid number!'
  end
  if delay <= 0 then
    return false, 'UpdateDelay must be greater than zero!'
  end
  if delay % 1 ~= 0 then
    return false, 'UpdateDelay must be an integer!'
  end

  return true
end
```

There is no upper bound for the delay.

### Severity and Impact Summary

Having upper bounds may help avoid mistakes such as malformatted timestamps.

### Recommendation

Consider adding additional bound checks where appropriate.

## Duplicate code

Finding ID: FYEO-OUTCOME-15
Severity: **Informational**
Status: **Remediated**

### Description

The `balanceById` and `balancesById` functions are identical.

### Proof of Issue

**File name:** src/marketModules/semiFungibleTokensValidation.lua
**Line number:** 69, 81

```
function semiFungibleTokensValidation.balanceById(msg, validPositionIds)
  if msg.Tags.Recipient then
    local success, err = sharedValidation.validateAddress(msg.Tags.Recipient,
'Recipient')
    if not success then return false, err end
  end
  return sharedValidation.validateItem(msg.Tags.PositionId, validPositionIds,
"PositionId")
end

function semiFungibleTokensValidation.balancesById(msg, validPositionIds)
  if msg.Tags.Recipient then
    local success, err = sharedValidation.validateAddress(msg.Tags.Recipient,
'Recipient')
    if not success then return false, err end
  end
  return sharedValidation.validateItem(msg.Tags.PositionId, validPositionIds,
"PositionId")
end
```

### Severity and Impact Summary

Code duplication should generally be avoided.

### Recommendation

Maintain just one function instead of having the same code twice.

## Events are issued without changes occuring

Finding ID: FYEO-OUTCOME-16
Severity: **Informational**
Status: **Remediated**

### Description

The `approveCreator` function creates notifications even if nothing changed as any creator can be approved any number of times. Furthermore, this function allows `unapproving` but this is not indicated by the docs. If that is intended functionality, make sure the name reflects this. I.e. use `updateCreatorApprovalStatus` as a name instead. The same applies to `approveCollateralToken` and `CPMMMethods:withdrawFee`.

### Proof of Issue

**File name:** src/marketFactoryModules/marketFactory.lua
**Line number:** 551

```
function MarketFactoryMethods:approveCreator(creator, approved, msg)
  self.approvedCreators[creator] = approved
  return self.approveCreatorNotice(creator, approved, msg)
end
```

**File name:** src/marketFactoryModules/marketFactory.lua
**Line number:** 650

```
function MarketFactoryMethods:approveCollateralToken(collateralToken, approved, msg)
  self.registeredCollateralTokens[collateralToken].approved = approved
  return self.approveCollateralTokenNotice(collateralToken, approved, msg)
end
```

**File name:** src/marketModules/cpmm.lua
**Line number:** 397

```
function CPMMMethods:withdrawFees(sender, onBehalfOf, cast, sendInterim, detached, msg)
  local feeAmount = self:feesWithdrawableBy(sender)
  if bint.__lt(0, bint(feeAmount)) then
    ...
  end
  if not cast then return self.withdrawFeesNotice(feeAmount, onBehalfOf, detached, msg)
end
end
```

### Severity and Impact Summary

The functions can be used to `unapprove` which is not indicated as intended functionality.

### Recommendation

Make sure these functions are implemented as required by specifications. It is generally advisable to not issue events if no data has changed.

## Mismatched check

Finding ID: FYEO-OUTCOME-17
Severity: **Informational**
Status: **Remediated**

### Description

Mismatch between check and comment `if not bint.__lt(totalFee, 1000) then 'Net fee must be less than or equal to 1000 bps'`.

### Proof of Issue

**File name:** src/marketModules/cpmmValidation.lua
**Line number:** 219

```
if not bint.__lt(totalFee, 1000) then
  return false, 'Net fee must be less than or equal to 1000 bps'
end
```

### Severity and Impact Summary

No security implication.

### Recommendation

Use `__le` or update the error message.

# 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