FYEO

Security Code Review of Launchpad Agent EVM

Bio

September 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

Bio engaged FYEO Inc. to perform a Security Code Review of Launchpad Agent EVM.

The assessment was conducted remotely by the FYEO Security Team. Testing took place on July 14 - September 17, 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-BIO-01 – Lack of uniqueness checks in proposeAgent() and incomplete event emission
- FYEO-BIO-02 – Missing `_disableInitializers()` in upgradeable contract
- FYEO-BIO-03 – Inconsistent role usage in `executeApplication`
- FYEO-BIO-04 – Missing `_disableInitializers()` in implementation contract
- FYEO-BIO-05 – Reducing `maxWeeks` can break `extend` logic
- FYEO-BIO-06 – Reentrancy and zero-amount check in `withdrawTax`, emits wrong event
- FYEO-BIO-07 – Uninitialized `AccessControl` and `Context` in `initialize`
- FYEO-BIO-08 – Voting units issued ignore time weight
- FYEO-BIO-09 – AdminUnlocked function does not return value

- FYEO-BIO-10 – Ambiguous `cliff` field assignment in vesting schedule
- FYEO-BIO-11 – Bound checks allow zero values
- FYEO-BIO-12 – Duplicate import of `SafeERC20`
- FYEO-BIO-13 – Ignoring return values from `EnumerableSet.add`/`remove`
- FYEO-BIO-14 – Incomplete validation and event emission in setImplementations
- FYEO-BIO-15 – Mismatch between AgentAddress struct order and return tuple in getAgentAddresses
- FYEO-BIO-16 – Missing `__ReentrancyGuard_init()` in initializer
- FYEO-BIO-17 – Missing events
- FYEO-BIO-18 – Missing zero check in setLaunchImplementation
- FYEO-BIO-19 – Missing zero checks and or bound checks
- FYEO-BIO-20 – No bounds check on `maxWeeks` in `initialize`
- FYEO-BIO-21 – Overly broad asset withdrawal after launch end
- FYEO-BIO-22 – Potential gas-exhaustion in `releaseAvailableTokensForHolder`
- FYEO-BIO-23 – Unbounded `applicationThreshold_` in `initFromApplication`
- FYEO-BIO-24 – Unbounded growth of `locks[]` risking DoS by gas exhaustion
- FYEO-BIO-25 – Unbounded tax rate changes in `setProjectTaxRates`
- FYEO-BIO-26 – `getPositions` may return a partially uninitialized array
- FYEO-BIO-27 – `getVestingSchedule` returns default for non-existent IDs

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 Launchpad Agent EVM. 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/bio-xyz/launchpad-agent-evm with the commit hash 0d0e4aa534e9a9dfaba5806e59ab2454192e534b.

Files included in the code review
```
launchpad-agent-evm/
└── src/
    ├── Launch/
    │   ├── Launch.sol
    │   ├── LaunchFactory.sol
    │   ├── LaunchLib.sol
    │   └── LaunchTypes.sol
    ├── Library/
    │   ├── SafeMath.sol
    │   └── UniswapV2Library.sol
    ├── interfaces/
    │   ├── IAgentFactory.sol
    │   ├── IAgentToken.sol
    │   ├── IAgentVestingToken.sol
    │   ├── IERC20Config.sol
    │   ├── ILaunch.sol
    │   ├── ILaunchFactory.sol
    │   ├── IStakedToken.sol
    │   ├── IUniswapV2Factory.sol
    │   ├── IUniswapV2Pair.sol
    │   ├── IUniswapV2Router01.sol
    │   └── IUniswapV2Router02.sol
    ├── AgentFactory.sol
    ├── AgentToken.sol
    ├── AgentVestingToken.sol
    ├── Execute.sol
    ├── StakedToken.sol
    └── veBIO.sol
```

Table 1: Scope

Technical Analyses and Findings

During the Security Code Review of Launchpad Agent EVM, we discovered:

- 2 findings with MEDIUM severity rating.
- 6 findings with LOW severity rating.
- 19 findings with INFORMATIONAL severity rating.

The following chart displays the findings by severity.

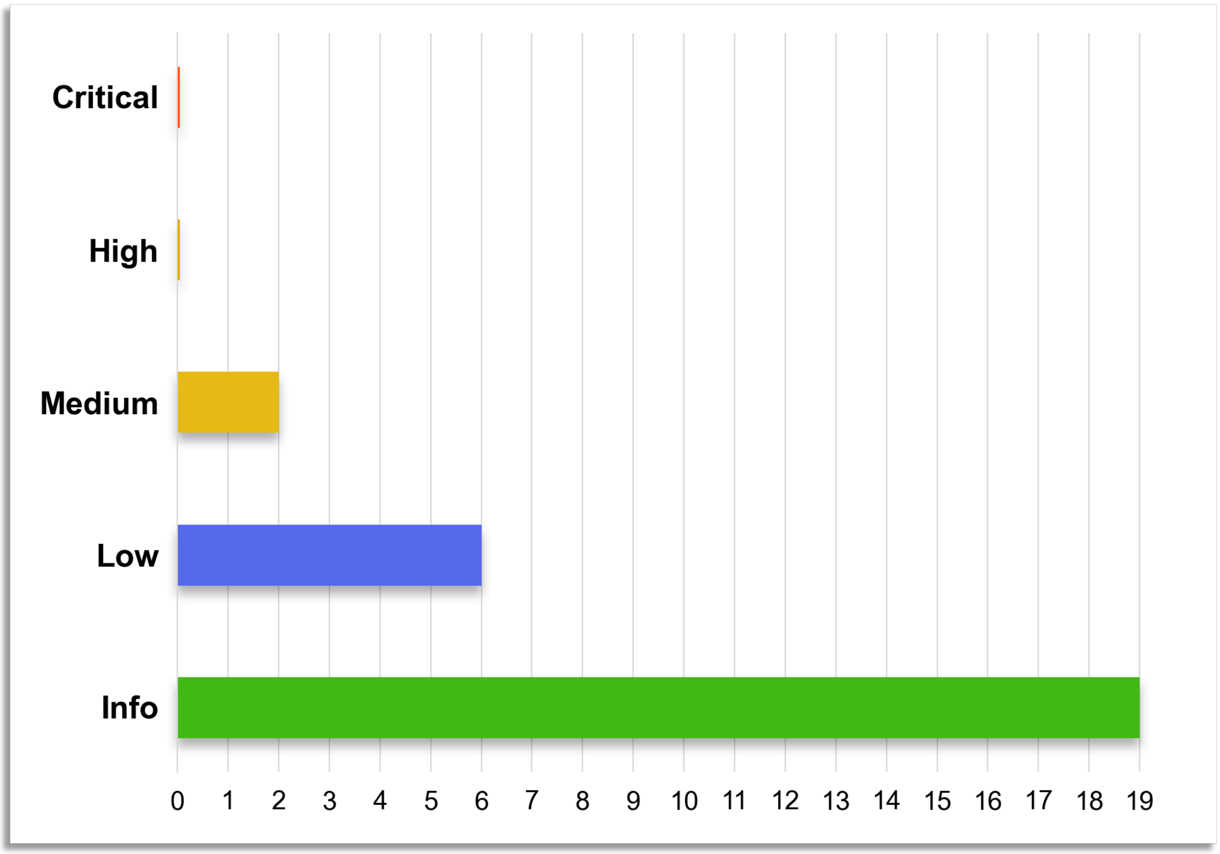

Figure 1: Findings by Severity

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	Status
FYEO-BIO-01	Medium	Lack of uniqueness checks in proposeAgent() and incomplete event emission	Remediated
FYEO-BIO-02	Medium	Missing `_disableInitializers()` in upgradeable contract	Remediated
FYEO-BIO-03	Low	Inconsistent role usage in `executeApplication`	Acknowledged
FYEO-BIO-04	Low	Missing `_disableInitializers()` in implementation contract	Remediated
FYEO-BIO-05	Low	Reducing `maxWeeks` can break `extend` logic	Remediated
FYEO-BIO-06	Low	Reentrancy and zero-amount check in `withdrawTax`, emits wrong event	Remediated
FYEO-BIO-07	Low	Uninitialized `AccessControl` and `Context` in `initialize`	Remediated
FYEO-BIO-08	Low	Voting units issued ignore time weight	Acknowledged
FYEO-BIO-09	Informational	AdminUnlocked function does not return value	Open
FYEO-BIO-10	Informational	Ambiguous `cliff` field assignment in vesting schedule	Open

FYEO-BIO-11	Informational	Bound checks allow zero values	Open
FYEO-BIO-12	Informational	Duplicate import of `SafeERC20`	Remediated
FYEO-BIO-13	Informational	Ignoring return values from `EnumerableSet.add`/`remove`	Remediated
FYEO-BIO-14	Informational	Incomplete validation and event emission in setImplementations	Remediated
FYEO-BIO-15	Informational	Mismatch between AgentAddress struct order and return tuple in getAgentAddresses	Remediated
FYEO-BIO-16	Informational	Missing `__ReentrancyGuard_init()` in initializer	Remediated
FYEO-BIO-17	Informational	Missing events	Remediated
FYEO-BIO-18	Informational	Missing zero check in setLaunchImplementation	Open
FYEO-BIO-19	Informational	Missing zero checks and or bound checks	Remediated
FYEO-BIO-20	Informational	No bounds check on `maxWeeks` in `initialize`	Open
FYEO-BIO-21	Informational	Overly broad asset withdrawal after launch end	Open

FYEO-BIO-22	Informational	Potential gas-exhaustion in `releaseAvailableTokensForHolder`	Remediated
FYEO-BIO-23	Informational	Unbounded `applicationThreshold_` in `initFromApplication`	Acknowledged
FYEO-BIO-24	Informational	Unbounded growth of `locks[]` risking DoS by gas exhaustion	Remediated
FYEO-BIO-25	Informational	Unbounded tax rate changes in `setProjectTaxRates`	Remediated
FYEO-BIO-26	Informational	`getPositions` may return a partially uninitialized array	Remediated
FYEO-BIO-27	Informational	`getVestingSchedule` returns default for non-existent IDs	Open

Table 2: Findings Overview

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

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 Launchpad Agent EVM implements a comprehensive token launch platform that enables projects to conduct fundraising campaigns through a structured, multi-phase process. The platform operates through a factory pattern where a central factory contract manages the creation and oversight of individual launch campaigns, while maintaining global parameters such as fee structures, contribution limits, campaign durations, and integration points with token minting infrastructure. Administrative roles control these system-wide settings and can pause operations or update parameters as needed, ensuring the platform remains flexible and manageable at scale.

When projects want to initiate a fundraising campaign, they interact with the factory to create a new launch by paying a small fee and specifying campaign-specific parameters like start times and initial configuration settings. The factory automatically assigns unique identifiers to each campaign and deploys dedicated smart contracts to handle the individual launch mechanics. This approach ensures that each campaign operates in isolation while benefiting from standardized infrastructure and shared security models established by the factory system.

During active campaign periods, participants can contribute to launches by staking utility tokens in exchange for participation points and tracked contribution amounts. The system carefully manages timing windows, only allowing participation between designated start and end times, while maintaining detailed records of each participant’s involvement. This staking mechanism creates a committed participant base while accumulating the resources needed for the campaign’s success, with all contributions tracked transparently on-chain for later processing and distribution.

Once campaigns conclude, authorized operators evaluate the results and trigger finalization processes that determine whether launches succeeded or failed based on predetermined criteria. Successful campaigns automatically trigger the creation of new governance tokens through integrated minting infrastructure, while failed campaigns are marked accordingly. This finalization step incorporates off-chain data through cryptographic proofs, allowing complex scoring and allocation logic to be verified on-chain without requiring expensive computation during the campaign period.

The platform’s final phase handles the distribution of results to participants through a claiming system that uses cryptographic proofs to verify individual allocations. Successful campaign participants can claim their proportional share of newly minted tokens along with any unused contributions, while failed campaign participants can recover their entire staked amounts. Administrative functions allow for the cleanup of remaining assets after campaigns conclude, while comprehensive role-based access controls ensure that only authorized parties can perform sensitive operations throughout the entire process, from campaign creation through final asset distribution.

Lack of uniqueness checks in proposeAgent() and incomplete event emission

Finding ID: FYEO-BIO-01
Severity: Medium
Status: Remediated

Description

The `proposeAgent()` function does not enforce uniqueness of agent names when storing applications. In `_executeApplication()`, the code assumes that agent names are unique keys when writing to the `agentAddresses` mapping. Without explicit checks, multiple agents with the same name will overwrite each other. Additionally, `NewPersona` event omits the `stakedToken`, leading to incomplete event logs.

Proof of Issue

File name: src/AgentFactory.sol
Line number: 252

```
$.agentAddresses[bytes(application.name)] =
    AgentAddress({veToken: veToken, agentToken: token, stakedToken: stakedToken});
```

Severity and Impact Summary

Overwriting existing agent records can lead to denial-of-service, broken references, and fund mismanagement. Missing `stakedToken` in the event reduces transparency and may hinder off-chain indexing.

Recommendation

Make sure names are unique. Normalize input strings (restrict to ASCII/readable characters) to avoid confusion. Update the `NewPersona` event to include `stakedToken`.

Missing `_disableInitializers()` in upgradeable contract

Finding ID: FYEO-BIO-02
Severity: Medium
Status: Remediated

Description

As an upgradeable contract using OpenZeppelin's `Initializable`, the implementation contract should disable initializers in its constructor to prevent it from being initialized directly.

Proof of Issue

File name: VeBIO.sol

```
no constructor
```

Severity and Impact Summary

Without disabling initializers, an attacker could call `initialize(...)` on the implementation.

Recommendation

Add a constructor in the implementation contract that calls `_disableInitializers()`.

Inconsistent role usage in `executeApplication`

Finding ID: FYEO-BIO-03
Severity: Low
Status: Acknowledged

Description

`executeApplication` allows both the proposer and any address with `WITHDRAW_ROLE` to execute; semantically this mixes “withdraw” power with “execute” power.

Proof of Issue

File name: AgentFactory.sol
Line number: 200

```
require(msg.sender == application.proposer || hasRole(WITHDRAW_ROLE, msg.sender), "Not
proposer");
```

Severity and Impact Summary

Granting `WITHDRAW_ROLE` the power to deploy new Agents may be unintended.

Recommendation

Clarify access policy—likely replace `WITHDRAW_ROLE` with a distinct `EXECUTE_ROLE`, or restrict execution to the proposer only.

Missing `_disableInitializers()` in implementation contract

Finding ID: FYEO-BIO-04
Severity: **Low**
Status: **Remediated**

Description

As an upgradeable contract inheriting from `Initializable`, the implementation must disable initializers in a constructor to prevent direct initialization of the implementation. The contract also mixes non upgradeable and upgradeable base classes. There is no init for the ReentrancyGuard.

Proof of Issue

File name: Launch.sol

Severity and Impact Summary

Without `_disableInitializers()`, an attacker could call `initialize(...)` on the implementation contract itself.

Recommendation

Add a constructor that calls `_disableInitializers()`.

Reducing `maxWeeks` can break `extend` logic

Finding ID: FYEO-BIO-05
Severity: Low
Status: Remediated

Description

If `setMaxWeeks` lowers `maxWeeks` below some existing `lock.numWeeks`, then a subsequent `extend` call will revert incorrectly because `(lock.numWeeks + numWeeks) > maxWeeks`.

Proof of Issue

File name: VeBIO.sol
Line number: 207

```
function setMaxWeeks(uint8 maxWeeks_) external onlyRole(ADMIN_ROLE) {
    maxWeeks = maxWeeks_;
}
```

Severity and Impact Summary

Adjusting `maxWeeks` downward can unintentionally lock out valid extensions for existing positions, degrading user experience.

Recommendation

Either forbid lowering `maxWeeks` below the current maximum `numWeeks` in active locks, or iterate all locks to clamp `numWeeks` and adjust `end` timestamps accordingly.

Reentrancy and zero-amount check in `withdrawTax`, emits wrong event

Finding ID: FYEO-BIO-06
Severity: Low
Status: Remediated

Description

`withdrawTax` performs the external token transfer before zeroing state and has no guard against zero `projectTaxPaid`.

Proof of Issue

File name: AgentToken.sol
Line number: 669

```
function withdrawTax() external onlyOwnerOrFactory {
    IERC20(address(this)).safeTransfer(projectTaxRecipient, projectTaxPaid);
    emit TaxWithdrawn(projectTaxRecipient, _balances[address(this)]);
    projectTaxPaid = 0;
}
```

Severity and Impact Summary

1. Transferring before zeroing state can be reentered to drain extra. This is an admin function however.
2. Calling with `projectTaxPaid == 0` emits misleading event.
3. Event reports wrong value.

Recommendation

Check `projectTaxPaid` is greater than 0. Copy `projectTaxPaid` first, then set it to zero and then do the withdraw on the copied value to avoid re-entrancy. Emit the correct value in the event.

Uninitialized `AccessControl` and `Context` in `initialize`

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

Description

The proxy `initialize(...)` calls only `__Pausable_init()`, but inherits non-upgradeable `AccessControl` and `Context` bases. Without their initializers, role-admin mappings and `_msgSender()` behavior may be incorrect or collide in storage.

Proof of Issue

File name: AgentFactory.sol
Line number: 99

```
function initialize(
    …
) public initializer {
    __Pausable_init();
    tokenImplementation = tokenImplementation_;
}
```

Severity and Impact Summary

Uninitialized base contracts can leave the factory without a valid admin, or with unpredictable storage, breaking access controls.

Recommendation

Use the upgradeable variants and chain all initializers:

```
__Context_init();
__AccessControl_init();
__Pausable_init();
```

Voting units issued ignore time weight

Finding ID: FYEO-BIO-08
Severity: Low
Status: Acknowledged

Description

On `stake`, the contract transfers voting units equal to `amount`, but ignores the lock duration (`numWeeks`) so longer locks aren't weighted more heavily in voting power.

Proof of Issue

File name: VeBIO.sol
Line number: 141

```
_transferVotingUnits(address(0), _msgSender(), amount);
```

Severity and Impact Summary

Ve-token design usually weights votes by both amount and duration; here, all stakes, even one week or auto-renew, get full weight, distorting governance.

Recommendation

Compute voting units proportional to `_calcValue(amount, numWeeks)` instead of raw `amount`.

AdminUnlocked function does not return value

Finding ID: FYEO-BIO-09
Severity: Informational
Status: Open

Description

The `adminUnlocked()` function is declared as a `view` (or intended to be a view) but does not return the stored boolean. There are at least two occurrences.

Proof of Issue

File name: src/StakedToken.sol
Location: `adminUnlocked()` function

```
function adminUnlocked() external view {
    _getStakedTokenStorage().adminUnlocked;
}
```

Severity and Impact Summary

This is a functional bug. Callers expect to read whether admin unlock is enabled. Currently the function returns nothing.

Recommendation

Change the function signature to explicitly return `bool` and `return` the stored value.

Ambiguous `cliff` field assignment in vesting schedule

Finding ID: FYEO-BIO-10
Severity: Informational
Status: Open

Description

The struct's `cliff` member is documented as a duration but is stored as `start + cliff`, i.e. a timestamp. This mismatch can confuse readers and lead to incorrect usage.

Proof of Issue

File name: AgentVeToken.sol
Line number: 317

```
vestingSchedules[vestingScheduleId] = VestingSchedule(
    _start + _cliff,
);
```

Severity and Impact Summary

This is purely a naming/documentation issue but may lead to misuse of the `cliff` field elsewhere.

Recommendation

Rename the struct field from `cliff` to `cliffTimestamp` (or store only the duration), and update comments accordingly.

Bound checks allow zero values

Finding ID: FYEO-BIO-11
Severity: Informational
Status: Open

Description

Several functions accept `0` for parameters where `0` is likely invalid and leads to incorrect behavior.

Proof of Issue

File name: src/StakedToken.sol
Functions: `initialize`, `getPositions`, `extend`

```
function initialize(address baseToken_, uint8 maxWeeks_, string memory name_, string
memory symbol_) ...
    $.maxWeeks = maxWeeks_; // <-- no bound check; can be 0
    ...

function getPositions(address account, uint256 start, uint256 count) ...
    Lock[] memory results = new Lock[](count); // count == 0 allowed
    ...

function extend(uint256 id, uint8 numWeeks) ...
    lock.numWeeks += numWeeks; // numWeeks == 0 allowed -> no-op
    ...
```

Severity and Impact Summary

1) Produces logically-invalid locks, locks end immediately.

2) Returning an empty array may be acceptable, but could cause surprising behavior.
3) No state change but still emits an event. Also may hide logic bugs in callers.

Recommendation

Create explicit input validation and clearer behavior.

Duplicate import of `SafeERC20`

Finding ID: FYEO-BIO-12
Severity: Informational
Status: Remediated

Description

The contract imports `SafeERC20` twice from the same path, which is redundant and may cause compiler warnings or confusion.

Proof of Issue

File name: VeBIO.sol
Line number: 7

```
import {SafeERC20} from "oz/contracts/token/ERC20/utils/SafeERC20.sol";
...
import {SafeERC20} from "oz/contracts/token/ERC20/utils/SafeERC20.sol";
```

Severity and Impact Summary

The duplication has no functional impact but should be cleaned up to prevent warnings and improve readability.

Recommendation

Remove the second `import {SafeERC20}` statement.

Ignoring return values from `EnumerableSet.add`/`remove`

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

Description

Several calls to `_liquidityPools.add(...)` and `.remove(...)` ignore the returned `bool` indicating if the set was changed, so re-adding or removing a non-existent pool silently fails.

Proof of Issue

File name: AgentToken.sol
Line numbers: 172, 281, 294

```
_liquidityPools.add();
_liquidityPools.remove();
```

Severity and Impact Summary

Items being removed may not exist. Items being added may exist.

Recommendation

Capture and require the return value to be as expected.

Incomplete validation and event emission in setImplementations

Finding ID: FYEO-BIO-14
Severity: Informational
Status: Remediated

Description

The `setImplementations()` function enforces uniqueness only between `token` and `veToken`, but not against `stakeToken`. This allows invalid configurations where tokens overlap. Additionally, the `ImplContractsUpdated` event does not include `stakeToken`, creating inconsistency between state and logs.

Proof of Issue

File name: src/AgentFactory.sol
Line number: 330

```
require(token != veToken, "Token and veToken cannot be the same");
...
emit ImplContractsUpdated(token, veToken);
```

Severity and Impact Summary

Overlapping token addresses may lead to broken logic, corrupted deployments, or token mismanagement. Missing event data reduces monitoring and auditability.

Recommendation

Add explicit checks to ensure all three addresses are distinct and nonzero. Update `ImplContractsUpdated` to include `stakeToken`.

Mismatch between AgentAddress struct order and return tuple in getAgentAddresses

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

Description

There is a mismatch between the order of fields in the `AgentAddress` struct and the order of return values in `getAgentAddresses()`. This inconsistency introduces confusion and increases the risk of incorrect assumptions by contract callers.

Proof of Issue

File name: src/AgentFactory.sol
Line number: 171

```
return ($.agentAddresses[key].agentToken, $.agentAddresses[key].veToken,
$.agentAddresses[key].stakedToken);
```

Severity and Impact Summary

Callers may assume the return tuple matches the struct ordering. Misinterpretation of fields can lead to logic bugs, incorrect token usage, or vulnerabilities in dependent contracts.

Recommendation

Return fields in the same order as the struct (`veToken, agentToken, stakedToken`). Alternatively, return a struct or named tuple for clarity. Add NatSpec comments documenting the return values explicitly.

Missing `__ReentrancyGuard_init()` in initializer

Finding ID: FYEO-BIO-16
Severity: Informational
Status: Remediated

Description

`AgentVeToken.initialize` calls `__Pausable_init()` and `__AccessControlDefaultAdminRules_init`, but never initializes the non-upgradeable `ReentrancyGuard` base. Under a proxy this leaves its internal lock state unset, potentially allowing reentrancy or storage collisions.

Proof of Issue

File name: AgentVeToken.sol
Line number: 188

```
function initialize(
    IERC20Metadata _underlyingToken,
    string memory _name,
    string memory _symbol,
    uint256 _coolDownPeriod
) public initializer {
    ...
```

Severity and Impact Summary

This contract mixes non upgradeable contracts with upgradeable contracts.

Recommendation

Use an upgradeable contract and call `__ReentrancyGuard_init()` to fully initialize inherited upgradeable guards.

Missing events

Finding ID: FYEO-BIO-17
Severity: Informational
Status: Remediated

Description

Changing the protocol’s settings and executing key actions does not always emit events.

Proof of Issue

File name: VeBIO.sol
Line number: 207

```
maxWeeks = maxWeeks_;
```

File name: AgentVeToken.sol
Line number: 384

```
function withdraw(uint256 amount) external nonReentrant onlyRole(DEFAULT_ADMIN_ROLE)
```

File name: AgentFactory.sol
Line number: 242

```
function setImplementations(address token, address veToken) public
onlyRole(DEFAULT_ADMIN_ROLE) {
    tokenImplementation = token;
    veTokenImplementation = veToken;
}
```

File name: AgentFactory.sol
Line number: 247

```
function setParams(address newRouter, address newTokenAdmin) public
onlyRole(DEFAULT_ADMIN_ROLE) {
    _uniswapRouter = newRouter;
    _tokenAdmin = newTokenAdmin;
}
```

File name: AgentFactory.sol
Line number: 252, 329

File name: `LaunchFactory.sol`
Line number: thirty-three (inside `setParams`)

```
function setParams(Params calldata p) external onlyRole(ADMIN_ROLE) {
    _setParams(p);
}
```

Severity and Impact Summary

Without an event, off-chain services cannot detect changes and users lack transparency.

Recommendation

Emit event in the appropriate places.

Missing zero check in setLaunchImplementation

Finding ID: FYEO-BIO-18
Severity: Informational
Status: Open

Description

The `setLaunchImplementation` setter allows setting the `launchImplementation` address with no validation. That permits assigning `address(0)` which can break later proxy deployments that rely on this value.

Proof of Issue

File name: `LaunchFactory.sol`

```
function setLaunchImplementation(address implementation) external onlyRole(ADMIN_ROLE) {
    LaunchFactoryStorage storage $ = _getLaunchFactoryStorage();
    $.launchImplementation = implementation;
}
```

Severity and Impact Summary

If set to `address(0)`, future proxy deployments will be nonfunctional.

Recommendation

Reject zero addresses and require the address to be a contract.

Missing zero checks and or bound checks

Finding ID: FYEO-BIO-19
Severity: Informational
Status: Remediated

Description

Throughout the codebase there are several missing zero checks.

Proof of Issue

File name: AgentVeToken.sol
Line number: 266

```
function vest(
    address _beneficiary,
    ...
```

File name: AgentToken.sol
Line number: 87

```
pairToken = integrationAddresses_[0...2];
```

File name: AgentToken.sol
Line number: 305

```
function setProjectTaxRecipient(address projectTaxRecipient_) ...
```

File name: AgentFactory.sol
Line number: 101

```
tokenImplementation = tokenImplementation_;
veTokenImplementation = veTokenImplementation_;
assetToken = assetToken_;
```

File name: AgentFactory.sol
Line number: 242

```
function setImplementations(address token, address veToken) public
onlyRole(DEFAULT_ADMIN_ROLE) {
    tokenImplementation = token;
    veTokenImplementation = veToken;
}
```

File name: AgentFactory.sol
Line number: 247

```
function setParams(address newRouter, address newTokenAdmin) public
onlyRole(DEFAULT_ADMIN_ROLE) {
    _uniswapRouter = newRouter;
    _tokenAdmin = newTokenAdmin;
}
```

File name: AgentFactory.sol
Line number: 252

```
function setTokenParams(
    uint256 lpSupply,
    uint256 projectBuyTaxBasisPoints,
    uint256 projectSellTaxBasisPoints,
    address projectTaxRecipient
) public onlyRole(DEFAULT_ADMIN_ROLE) {
    require(lpSupply > 0, "Invalid supply");
    _lpSupply = lpSupply;
    _tokenTaxParams = abi.encode(projectBuyTaxBasisPoints, projectSellTaxBasisPoints,
projectTaxRecipient);
}
```

This should check these are valid BPS bounds.

File name: Launch.sol
Line number: 114

```
require(bytes(params.launchName).length > 0, "Invalid launch name");
require(bytes(params.launchTicker).length > 0, "Invalid launch ticker");
```

Add more specific length checks.

Severity and Impact Summary

Misconfiguration or malicious input could break core logic or lock tokens.

Recommendation

Add checks to validate against the zero address.

No bounds check on `maxWeeks` in `initialize`

Finding ID: FYEO-BIO-20
Severity: Informational
Status: Open

Description

The initializer sets `maxWeeks = maxWeeks_` without any upper or lower bounds checks. An arbitrarily large value could lead to arithmetic overflows or logic errors elsewhere.

Proof of Issue

File name: VeBIO.sol
Line number: 55, 207

```
maxWeeks = maxWeeks_;
```

Severity and Impact Summary

Severity: Medium An attacker (or misconfigured deployer) could set `maxWeeks_ = 0` (breaking all locks) or an extremely large value (risking overflow in time calculations).

Recommendation

Add a sanity check.

Overly broad asset withdrawal after launch end

Finding ID: FYEO-BIO-21
Severity: Informational
Status: Open

Description

After a launch ends, the `withdrawLeftAssetsAfterFinalized` function lets any admin withdraw *any* ERC-20 token and amount up to the full contract balance. There is no whitelist or distinction between “unsold” or “residual” assets and participants’ deposited funds or agent tokens that users still need to claim.

Proof of Issue

File name: Launch.sol
Line number: 277

```
function withdrawLeftAssetsAfterFinalized(address to, address token, uint256 amount)
    external
    onlyRole(DEFAULT_ADMIN_ROLE)
    nonReentrant
    whenEnded
{
    require(token != address(0), "Invalid token address");
    require(amount <= IERC20(token).balanceOf(address(this)), "Insufficient balance to
withdraw");

    IERC20(token).safeTransfer(to, amount);

    emit AssetsWithdrawn(launchId, to, token, amount);
}
```

Severity and Impact Summary

An admin could prematurely or maliciously withdraw participant contributions (BIO), agent tokens, or other crucial assets immediately after `endTime`, potentially preventing users from claiming what they are owed.

Recommendation

Make sure this can not be abused by a malicious admin.

Potential gas-exhaustion in `releaseAvailableTokensForHolder`

Finding ID: FYEO-BIO-22
Severity: **Informational**
Status: **Remediated**

Description

Iterates over all of a holder’s schedules in a single transaction. If a user has many schedules, this loop can exceed block gas limits and become unusable.

Proof of Issue

File name: AgentVeToken.sol
Line number: 428

```
for (uint256 i = 0; i < vestingScheduleCount; i++) {
    ...
}
```

Severity and Impact Summary

Large numbers of small schedules will eventually render this function uncallable, locking vested tokens.

Recommendation

Impose a per-user cap on schedules.

Unbounded `applicationThreshold_` in `initFromApplication`

Finding ID: FYEO-BIO-23
Severity: Informational
Status: Acknowledged

Description

`initFromApplication(...)` takes a caller-provided `applicationThreshold_` and deposits it, but doesn't check against a global `applicationThreshold`, allowing zero or arbitrary amounts.

Proof of Issue

File name: AgentFactory.sol
Line number: 279

```
require(IERC20(assetToken).balanceOf(sender) >= applicationThreshold_, "Insufficient
asset token");
…
IERC20(assetToken).safeTransferFrom(sender, address(this), applicationThreshold_);
```

Severity and Impact Summary

A malicious launcher could bypass threshold logic or deposit zero, breaking economic guarantees.

Recommendation

Make sure this is implemented according to requirements.

Unbounded growth of `locks[]` risking DoS by gas exhaustion

Finding ID: FYEO-BIO-24
Severity: **Informational**
Status: **Remediated**

Description

Users can push unlimited `Lock` structs into their `locks[account]` array. Although a `MAX_POSITIONS` constant exists, it's never enforced, leading to potential out-of-gas or denial of service when iterating.

Proof of Issue

File name: VeBIO.sol
Line number: 139

```
locks[_msgSender()].push(lock);
```

Severity and Impact Summary

An attacker or even a benign user might exceed practical limits, making any view or interaction (e.g., `balanceOfAt`, iteration) revert due to gas limits.

Recommendation

Enforce a per user cap.

Unbounded tax rate changes in `setProjectTaxRates`

Finding ID: FYEO-BIO-25
Severity: **Informational**
Status: **Remediated**

Description

Although the comment says rates can only decrease, there is no check preventing increases or exceeding `BP_DENOM`.

Proof of Issue

File name: AgentToken.sol
Line number: 318

```
function setProjectTaxRates(uint16 newBuy, uint16 newSell)  {

    projectBuyTaxBasisPoints = newBuy;
    projectSellTaxBasisPoints = newSell;
```

Severity and Impact Summary

Admins could raise tax rates arbitrarily (up to 100%), harming users unexpectedly.

Recommendation

Enforce bounds. Update the documentation to reflect actual functionality.

`getPositions` may return a partially uninitialized array

Finding ID: FYEO-BIO-26
Severity: Informational
Status: Remediated

Description

The function allocates an array of length `count`, but if `locks[account].length < start + count`, some slots remain unassigned, leaving zeroed-out `Lock` structs.

Proof of Issue

File name: VeBIO.sol
Line number: 67

```
Lock[] memory results = new Lock[](count);
```

Severity and Impact Summary

Consumers of this view may misinterpret zeroed entries as real locks, leading to confusion or incorrect off-chain accounting.

Recommendation

Either return a smaller array sized to the actual number of returned locks, or (2) require `start + count <= locks[account].length` and revert otherwise.

`getVestingSchedule` returns default for non-existent IDs

Finding ID: FYEO-BIO-27
Severity: Informational
Status: Open

Description

Calling `getVestingSchedule(bytes32)` on an unknown ID silently returns a zeroed `VestingSchedule`, rather than reverting, potentially misleading callers.

Proof of Issue

File name: AgentVeToken.sol
Line number: 456

```
function getVestingSchedule(bytes32 vestingScheduleId) public view returns
(VestingSchedule memory) {
    return vestingSchedules[vestingScheduleId];
}
```

File name: AgentFactory.sol
Line number: 114

```
function getApplication(uint256 proposalId) public view returns (Application memory) {
    return _applications[proposalId];
}
```

Severity and Impact Summary

Consumers may be confused.

Recommendation

Add a check and revert if `vestingSchedules[vestingScheduleId].duration == 0`.

Our Process

Methodology

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

Kickoff
Ramp-up
Review
Report
Verify

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.