FANTIUM SECURITY AUDIT REPORT

May 4, 2023

MixBytes()

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only. The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of the Client. If you are not the intended recipient(s) of this document, please note that any disclosure, copying or dissemination of its content is strictly forbidden.

1.2 Security Assessment Methodology

A group of auditors are involved in the work on the audit. The security engineers check the provided source code independently of each other in accordance with the methodology described below:

1. Project architecture review:

- Project documentation review.
- General code review.
- Reverse research and study of the project architecture on the source code alone.

Stage goals
- Build an independent view of the project's architecture.
- Identifying logical flaws.

2. Checking the code in accordance with the vulnerabilities checklist:

- Manual code check for vulnerabilities listed on the Contractor's internal checklist. The Contractor's checklist is constantly updated based on the analysis of hacks, research, and audit of the clients' codes.
- Code check with the use of static analyzers (i.e Slither, Mythril, etc).

Stage goal

Eliminate typical vulnerabilities (e.g. reentrancy, gas limit, flash loan attacks etc.).

3. Checking the code for compliance with the desired security model:

- Detailed study of the project documentation.
- Examination of contracts tests.
- Examination of comments in code.
- Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit.
- Exploits PoC development with the use of such programs as Brownie and Hardhat.

Stage goal

Detect inconsistencies with the desired model.

4. Consolidation of the auditors' interim reports into one:

- Cross check: each auditor reviews the reports of the others.
- Discussion of the issues found by the auditors.
- Issuance of an interim audit report.

Stage goals

- Double-check all the found issues to make sure they are relevant and the determined threat level is correct.
- Provide the Client with an interim report.

5. Bug fixing & re-audit:

- The Client either fixes the issues or provides comments on the issues found by the auditors. Feedback from the Customer must be received on every issue/bug so that the Contractor can assign them a status (either "fixed" or "acknowledged").
- Upon completion of the bug fixing, the auditors double-check each fix and assign it a specific status, providing a proof link to the fix.
- A re-audited report is issued.

Stage goals

- Verify the fixed code version with all the recommendations and its statuses.
- Provide the Client with a re-audited report.

6. Final code verification and issuance of a public audit report:

- The Customer deploys the re-audited source code on the mainnet.
- The Contractor verifies the deployed code with the re-audited version and checks them for compliance.
- If the versions of the code match, the Contractor issues a public audit report.

Stage goals

- Conduct the final check of the code deployed on the mainnet.
- Provide the Customer with a public audit report.

Finding Severity breakdown

All vulnerabilities discovered during the audit are classified based on their potential severity and have the following classification:

Severity	Description
Critical	Bugs leading to assets theft, fund access locking, or any other loss of funds.
High	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.
Medium	Bugs that can break the intended contract logic or expose it to DoS attacks, but do not cause direct loss funds.
Low	Bugs that do not have a significant immediate impact and could be easily fixed.

Based on the feedback received from the Customer regarding the list of findings discovered by the Contractor, they are assigned the following statuses:

Status	Description
Fixed	Recommended fixes have been made to the project code and no longer affect its security.
Acknowledged	The Customer is aware of the finding. Recommendations for the finding are planned to be resolved in the future.

1.3 Project Overview

Fantium is the platform where users can invest in athletes' NFTs. The project contains an NFT Contract and a special controller to validate users and mint tokens. There is an excessive centralization in the protocol due to the `trustedForwarder` logic. Every user should trust this address because `trustedForwarder` can call any function on behalf of any user. Also, there is an upgrading logic in the contract which should be controlled by a DAO or multisig with a secure amount of signers. In addition, the platform manager has unlimited rights in updating the parameters of collections and reward payouts, so it must be a multisig.

1.4 Project Dashboard

Project Summary

Title	Description
Client	Fantium
Project name	Fantium
Timeline	March 6 2023 - May 1 2023
Number of Auditors	4

Project Log

Date	Commit Hash	Note
06.03.2023	a2d126453c1105028f12277b8f342d2cdbf01a77	Commit for the audit
10.04.2023	c622b4c35132167bacc22cf76656d80715edb5fd	Commit for the reaudit
17.04.2023	b4acd978e25289becc640f75eb5921e760b83720	Commit for the reaudit 2
19.04.2023	0ad51f891f69eeb67d1d5e3ace824b9f8157952b	Commit for the reaudit 3

Project Scope

The audit covered the following files:

File name	Link
FantiumNFTV3.sol	FantiumNFTV3.sol
FantiumClaimingV1.sol	FantiumClaimingV1.sol
FantiumUserManager.sol	FantiumUserManager.sol
TokenVersionUtil.sol	TokenVersionUtil.sol

Deployments

File name	Contract deployed on mainnet	Comment
FantiumNFTV3	0x167fE0F3aaa94698C8B9cf4D4770D7fB4b30f454	Proxy for the contract 0x2b98132E7cfd88C5D854d64f436372838A9BA49d
FantiumClaimingV1	0xc609B07dA3e23eAD4D41ebA31694880F4b5945e1	Proxy for the contract 0x534db6CE612486F179ef821a57ee93F44718a002

File name	Contract deployed on mainnet	Comment
FantiumUserManager	0x41aE20b0087549253eeAb91955219E919242896c	Proxy for the contract 0xBf1cabD8C250D2ecF0Bf0D0f04ffF79f3F2903C0

1.5 Summary of findings

Severity	# of Findings
Critical	4
High	15
Medium	25
Low	31

ID	Name	Severity	Status
C-1	An incorrect version calculation will lead to a token loss for a user	Critical	Fixed
C-2	There are no checks that an athlete sent funds to the FantiumClaimingV1 contract	Critical	Fixed
C-3	An unlimited claim	Critical	Fixed
C-4	A possible transfer of funds of another athlete	Critical	Fixed
H-1	Some distribution parameters cannot be updated after funds sending	High	Fixed
H-2	Some tokens could be left on the contract	High	Fixed
H-3	TransferFrom from `address(this)` will not work for most of the tokens	High	Fixed
H-4	Not enough checks for collections shares	High	Fixed
H-5	`_athleteSecondarySalesBPS` and `_fantiumSecondarySalesBPS` should be restricted	High	Fixed

H-6	`maxInvocations` should be limited	High	Fixed
H-7	`trustedForwarder` can be used to impersonate any account or contract	High	Fixed
H-8	An incorrect expression inside `require` leads to a revert	High	Fixed
H-9	Not checking the ERC20 transfer result on depositing funds by an athlete	High	Fixed
H-10	Funds sent by an athlete to a closed distribution are impossible to return	High	Fixed
H-11	Updating a token version can cause a jump of tokens between collections	High	Fixed
H-12	Inconsistent shares upgrade	High	Fixed
H-13	An incorrect check	High	Fixed
H-14	It's possible to top up a closed distribution	High	Fixed
H-15	Distribution parameters shouldn't be updated after the first claim	High	Fixed
M-1	Incorrect event emitting	Medium	Fixed
M-2	Incorrect input data could break the collection	Medium	Fixed
M-3	Unsafe transfer	Medium	Fixed
M-4	`nextCollectionId` should be limited	Medium	Fixed
M-5	Collection shares should be limited	Medium	Fixed
M-6	`maxInvocations` cannot be reduced during an update	Medium	Fixed
M-7	Modifier `onlyValidCollectionId()` is missing	Medium	Fixed
M-8	Distribution logic is too flexible	Medium	Fixed

M-9	`fantiumFeeBPS` should be restricted	Medium	Fixed
M-10	An unsafe transfer	Medium	Fixed
M-11	Check for the collection existence is missing	Medium	Fixed
M-12	Possible division by zero	Medium	Fixed
M-13	Distribution closing shouldn't be allowed during the active distribution	Medium	Acknowledged
M-14	The collection existence check is missing	Medium	Fixed
M-15	Variable shadowing	Medium	Fixed
M-16	The NFT contract allows multiple ways to avoid KYC	Medium	Acknowledged
M-17	Hardcoded values for amounts of ERC20 tokens	Medium	Fixed
M-18	A manager could transfer funds to the Claiming contract instead of the athlete	Medium	Acknowledged
M-19	Claiming could be blocked	Medium	Fixed
M-20	The check can be manipulated	Medium	Fixed
M-21	Incorrect work with decimals	Medium	Fixed
M-22	Unnecessary check reverts function call.	Medium	Fixed
M-23	Anyone can block the payout token by changing its amount by 1 wei	Medium	Fixed
M-24	A missed requirement	Medium	Fixed
M-25	`takeClaimingSnapshot()` call will lead to an inconsistent state	Medium	Acknowledged
L-1	Possible incorrect event emitting	Low	Fixed
L-2	A function can be called with `amount == 0`	Low	Fixed

L-3	Minting insolvency	Low	Fixed
L-4	Unnecessary checks	Low	Fixed
L-5	A zero address check	Low	Fixed
L-6	A front-run risk	Low	Acknowledged
L-7	An unchecked timestamp	Low	Acknowledged
L-8	Events missing	Low	Fixed
L-9	Bad UX on claiming	Low	Acknowledged
L-10	An inconvenient token update	Low	Acknowledged
L-11	Initialization front-running	Low	Acknowledged
L-12	No `forwarder` address check	Low	Acknowledged
L-13	Bad work with decimals	Low	Fixed
L-14	An unnecessary parameter	Low	Fixed
L-15	Unused logic	Low	Acknowledged
L-16	Unchecked range for time	Low	Acknowledged
L-17	Unnecessary checks	Low	Fixed
L-18	Not checking the length on `_addresses` and `_increaseAllocations`	Low	Fixed
L-19	Unnecessary casting to `uint256`	Low	Fixed
L-20	`onlyManager()` and `onlyRole(PLATFORM_MANAGER_ROLE)` have the same logic	Low	Fixed
L-21	Equation can be simplified	Low	Fixed
L-22	Bad name for the function	Low	Fixed

L-23	A function can be optimized	Low	Fixed
L-24	Bad pause management	Low	Fixed
L-25	A limit of the distribution amount as one billion is not correct	Low	Fixed
L-26	Version calculation can be simplified	Low	Fixed
L-27	An unnecessary internal function onlyRole modifier	Low	Fixed
L-28	`payOutToken` and `payoutToken` variables differ only in one symbol.	Low	Fixed
L-29	Functions `topUpDistributionEvent()` and `addDistributionAmount()` have duplicated functionality.	Low	Fixed
L-30	The `maxInvocations` parameter check	Low	Acknowledged
L-31	An unnecessary check	Low	Fixed

1.6 Conclusion

During the audit process 4 CRITICAL, 15 HIGH, 25 MEDIUM, and 31 LOW severity findings were spotted. After working on the reported findings, all of them were acknowledged or fixed by the client.

2. FINDINGS REPORT

2.1 Critical

C-1	An incorrect version calculation will lead to a token loss for a user
Severity	Critical
Status	Fixed in c622b4c3

Description

A token version is calculated incorrectly (the version is in range [10_000:990_000]) which means that a user will lose a token from the current collection and receive a token from another collection (which probably hasn't existed yet).
TokenVersionUtil.sol#L23-L24

Recommendation

We recommend changing the calculation of the token version logic in the `getTokenInfo()` function in the `TokenVersionUtil` library.

C-2	There are no checks that an athlete sent funds to the FantiumClaimingV1 contract
Severity	Critical
Status	Fixed in c622b4c3

Description

There are no checks that an athlete has sent tokens to the contract (`amountPaidIn` is not checked). So, users from the collections that are not able to claim funds will be able to claim them from other distributions. It can be extremely dangerous if some of the athletes do not send funds to the contract on time.
FantiumClaimingV1.sol#L405

Recommendation

We recommend adding a simple check: `require(distributionEvents[_distributionEventId].amountPaidIn, "Unable to claim");`.

Client's commentary

> Client: Resolved by adding the proposed require statement to the `claim` function. Also added a further check such if `distributionEvent` exists.

C-3	An unlimited claim
Severity	Critical
Status	Fixed in c622b4c3

Description

An unnecessary check was implemented instead of updating the mapping here:
FantiumClaimingV1.sol#L456-L458
This allows an unlimited claim for a user, so anybody with an NFT could drain the contract.

Recommendation

We recommend adding a change from `==` to `=`.

Client's commentary

> Client: Updated by changing '==' to '=' in the `claim` function.

C-4	A possible transfer of funds of another athlete
Severity	Critical
Status	Fixed in c622b4c3

Description

There is no check that an athlete has already transferred funds to the contract here: FantiumClaimingV1.sol#L624-L652
It means that the manager by mistake can transfer the funds of another athlete to the athlete that asked to close their distribution.

Recommendation

We recommend adding a simple check: `require(distributionEvents[_distributionEventId].amountPaidIn, "Unable to close");`.

Client's commentary

> Client: Resolved by adding the proposed check to make sure the `amountPaidIn` is true in the closingDistributionEvent function.

2.2 High

H-1	Some distribution parameters cannot be updated after funds sending
Severity	High
Status	Fixed in b4acd978

Description

The updating of `tournamentDistributionAmount`, `otherDistributionAmount` and `collectionIds` could lead to the situation when some users would be unable to claim their funds, because the funds had already been claimed by other users.
FantiumClaimingV1.sol#L300-L302
FantiumClaimingV1.sol#L315

Recommendation

We recommend adding the following check: `require(!distributionEvents[_distributionEventId].amountPaidIn, "Cannot update parameters");`.

Client's commentary

> Client: Resolved with a new compute logic that only allows minted tokens to claim. Also it calculates amounts to be distributed based on the share each token has and how many tokens were minted at the point of snapshot.
> MB: `triggerClaimingSnapshot(_id)` should be called in the `updateDistributionEventCollectionIds()` function.
> Client: Added to the `update()` function.

H-2	Some tokens could be left on the contract
Severity	High
Status	Fixed in b4acd978

Description

The manager can update the token address before all tokens were distributed among fans. This can block some tokens on the contract and fans can receive tokens from other distributions.
FantiumClaimingV1.sol#L614

Recommendation

We recommend adding a check that the transferred tokens were distributed or closed by athletes.

Client's commentary

> Client: Added a check if the balance of the contracts payoutToken is 0.
> MB: The check should be: `IERC20Upgradeable(payoutToken).balanceOf(address(this)) == 0,`.
> Client: Corrected.

H-3	TransferFrom from `address(this)` will not work for most of the tokens
Severity	High
Status	Fixed in c622b4c3

Description

Here `transferFrom(address(this),...)` is used which will not work with most of the tokens without calling `approve`.
FantiumClaimingV1.sol#L647-L651

Recommendation

We recommend using `SafeTransfer` instead of `TransferFrom`.

H-4	Not enough checks for collections shares
Severity	High
Status	Fixed in c622b4c3

Description

There are not enough checks on the collections shares (tournament and others) which could lead to an unfair distribution of rewards for users. If `sum(tournamentTokenShare1e7_i) > 1e7 * tournamentBPS / 10_000`, then not all users will be able to claim rewards.
FantiumClaimingV1.sol#L225-L232

Recommendation

We recommend adding two checks:
`sum(tournamentTokenShare1e7_i) <= 1e7 * tournamentBPS / 10_000`
`sum(otherTokenShare1e7_i) <= 1e7 * otherBPS / 10_000`.

Client's commentary

> Client: Not relevant anymore as the calculation logic has changed. We are not using the totalBPS anymore but calculate the distributionAmount based on tokens minted.
> MB: After updates this issue does not exist anymore.

H-5	`_athleteSecondarySalesBPS` and `_fantiumSecondarySalesBPS` should be restricted
Severity	High
Status	Fixed in c622b4c3

Description

`_athleteSecondarySalesBPS` and `_fantiumSecondarySalesBPS` should be restricted because they will be used in other contracts to calculate royalties with base point = 1 / 10_000, so if `_athleteSecondarySalesBPS + _fantiumSecondarySalesBPS > 10_000`, the royalty will be > 100%. Due to this all transactions with royalty payments will revert.
FantiumNFTV3.sol#L526-L527
FantiumNFTV3.sol#L544-L545
FantiumNFTV3.sol#L658-L659
FantiumNFTV3.sol#L752-L753

Recommendation

We recommend adding the following check: `_athleteSecondarySalesBPS + _fantiumSecondarySalesBPS <= 10_000`.

H-6	`maxInvocations` should be limited
Severity	High
Status	Fixed in c622b4c3

Description

TokenId contains collactionId, versionId and tokenNumber. `maxInvocations` is a parameter that determines the max tokenNumber. Due to using versionId, tokenNumber should be < 10_000, otherwise the versionId logic would be broken.
FantiumNFTV3.sol#L528
FantiumNFTV3.sol#L691

Recommendation

We recommend adding the following check: `maxInvocations < 10_000`.

Client's commentary

> MB: The check is inconsistent: FantiumNFTV3.sol#L452 FantiumNFTV3.sol#L623.
> Client: Adjusted.

H-7	`trustedForwarder` can be used to impersonate any account or contract
Severity	High
Status	Fixed in 0ad51f89

Description

There is an overriden `_msgSender()` function FantiumNFTV3.sol#L993 which allows the `trustedForwarder` address owner to specify any address being used as `sender`. This function is used at some of the key parts of the protocol. For example, in the `upgradeTokenVersion` function - FantiumNFTV3.sol#L790. Someone's address and allowance can also be used during mint - FantiumNFTV3.sol#L350. In case `trustedForwarder` is compromised, it can lead to the funds loss.

Recommendation

We recommend introducing a multisig wallet on the `trustedForwarder` account or disabling the usage of the `_msgSender()` function on the key parts of the protocol.

Client's commentary

> Client: A trusted forwarder will be used to implement a relay that users without gas can use to mint and claim. This will help us onboard more web2 users. The trusted forwarder will be the address of the relay contract used for those actions. Do you have a recommendation on what we should change around security for that?
> MB: `PLATFORM_MANAGER_ROLE` still can be compromised by `trustedForwarder`, so we recommend using `msg.sender` in checks for the manager like this: FantiumNFTV3.sol#L130.
> Client: Removed it from all hasRole calls.
> MB: In `onlyRole` calls `AccessControlUpgradeable` logic is used where `_msgSender()` is used, so `trustedForwarder` still can bypass this check.
> Client: Added a custom onlyPlatformManager and onlyUpgrader modified that check hasRole() with msg.sender and replaced all default modifiers.

H-8	An incorrect expression inside `require` leads to a revert
Severity	High
Status	Fixed in c622b4c3

Description

There is a `closeDistribution` function - FantiumClaimingV1.sol#L624. It is used by the protocol manager to close a distribution event and transfer the remaining funds to the athlete. But the first require here FantiumClaimingV1.sol#L632 uses an assignment operator instead of a logic operator. It always assigns a false `closed` parameter on a distribution event and uses `false` as a `require` argument that leads to an unconditional revert. There is also a second `require` check FantiumClaimingV1.sol#L636 which uses an incorrect operator too.

Recommendation

We recommend using a logic operator == inside the mentioned `require` statements.

Client's commentary

> Client: Resolved by correcting operators.

H-9	Not checking the ERC20 transfer result on depositing funds by an athlete
Severity	High
Status	Fixed in b4acd978

Description

There is an `addDistributionAmount` function which transfers funds from an athlete to the protocol. It uses the `transferFrom` function here - FantiumClaimingV1.sol#L277. If `payoutToken` which returns false/true instead of reverting on an unsuccessful transfer would be used there, then an athlete would be able to mark the event as `amountPaidIn` without depositing any tokens.

Recommendation

We recommend using `safeTranferFrom` from the OpenZeppelin `safeERC20` implementation.

Client's commentary

> MB: There is still a problem with fee on transfer tokens. We also recommend adding a check for how many tokens were transferred via calling `balanceOf` before and after the `safeTrasnferFrom` call.
> Client: Added the balanceOf check before and after to the transfer function calls.
> MB: We do not recommend using an additional `balanceOf()` check in such cases where tokens are transferred from the Claiming contract. FantiumClaimingV1.sol#L670, FantiumClaimingV1.sol#L801
> Client: Removed.

H-10	Funds sent by an athlete to a closed distribution are impossible to return
Severity	High
Status	Fixed in c622b4c3

Description

It's possible for an athlete to send funds to a closed distribution with `FantiumClaimingV1.addDistributionAmount()` if they were not sent before (FantiumClaimingV1.sol#L253). After that it will be impossible to return them to the athlete (it's possible to call`FantiumClaimingV1.closeDistribution()` only for non-closed distributions). For example, it can happen if the incorrect distribution was created by the manager, the manager closed the distribution and recreated it. And by mistake the athlete called `FantiumClaimingV1.addDistributionAmount()` with an id of the closed one.

Recommendation

It's recommended to add a check to `FantiumClaimingV1.addDistributionAmount()` that the distribution is not closed.

Client's commentary

> Client: Resolved by adding a require statement to the addDistributionAmount function.

H-11	Updating a token version can cause a jump of tokens between collections
Severity	High
Status	Fixed in c622b4c3

Description

One collection can store only 1_000_000 token ids and one version can store maximum 10_000 of token ids that means a token version can be updated only 100 times. Moreover, updating the token version the 100th time will cause a jump of this token to the next collections.

Recommendation

We recommend expanding the number of updates for one token without limiting further the `maxInvocation` parameter in one collection and adding a check that `tokenVersionUpgrade` doesn't jump between collections FantiumNFTV3.sol#L812.

Client's commentary

> Client: Added the check that the version is below 100 at the recommended spot.

H-12	Inconsistent shares upgrade
Severity	High
Status	Fixed in b4acd978

Description

After shares update in the NFT contract FantiumNFTV3.sol#L638-L640 `tournamentDistributionAmount` and `otherDistributionAmount` should be updated for all distributions for the specific collection, otherwise, there will be inconsistency in the calculations of the amount for the claim.

Recommendation

We recommend saving the share value in the distribution event and using it in the calculations of the claim amount.

Client's commentary

> Client: If we are pulling also the share values into the contract, the calculation for a payout per token can be done when a snapshot is taken. It makes the contract at least a bit more efficient. I have changed the logic to store the claim per token of a collection within an additional struct. That means at function call claim, no additional calculation needs to be done which could lead to wrong payouts for certain users due to a change in some values in the NFT contract.
> MB: Function `takeClaimingSnapshot` FantiumClaimingV1.sol#L678-L691 can cause an inconsistent state, so we recommend removing it.

H-13	An incorrect check
Severity	High
Status	Fixed in b4acd978

Description

There is an incorrect check in the `updateDistribtionTotalEarningsAmounts()` function FantiumClaimingV1.sol#L326-L327. Instead of a check for `_totalTournamentEarnings` and `_totalOtherEarnings` there should be a check for `tournamentDistributionAmount` and `otherDistributionAmount` after calling the `triggerClaimingSnapshot` function. Otherwise, the manager could reduce `tournamentDistributionAmount` and `otherDistributionAmount` by mistake and block the distribution (the athlete will not be able to add tokens and will also close the distribution).

Recommendation

We recommend adding a check for `tournamentDistributionAmount` and `otherDistributionAmount` after calling the `triggerClaimingSnapshot` function. Also, the same check should be added to the `updateDistributionEventCollectionIds` function.

Client's commentary

> Client: Added to both functions and corrected the check.

H-14	It's possible to top up a closed distribution
Severity	High
Status	Fixed in 0ad51f89

Description

If the manager calls by mistake `updateDistribtionTotalEarningsAmounts()` or `updateDistributionEventCollectionIds()` for a closed distribution, it will be topped up using the athlete's funds. These funds will be stuck because it's impossible to close the already closed distribution again.
FantiumClaimingV1.sol#L318
FantiumClaimingV1.sol#L342

Recommendation

We recommend adding a check to `updateDistribtionTotalEarningsAmounts()` and `updateDistributionEventCollectionIds()`: `require(!distributionEvents[_id].closed, "FantiumClaimingV1: distribution already closed");`.

Client's commentary

> Client: Added to both function calls.

H-15	Distribution parameters shouldn't be updated after the first claim
Severity	High
Status	Fixed in 0ad51f89

Description

If the athlete/manager decides to update the parameters of the distribution event after the tokens have been sent from the claiming contract, it may lead to an unequal distribution of tokens among users. This outcome is inherent in the design. For example, updating the collection IDs while the claiming process is in progress implies that an individual from the previous collection could claim the rewards for themselves and expend some funds from the distribution. Consequently, someone from the new collection may be unable to claim rewards. Updating the `distributionTotalEarningAmounts`signifies that a person who has already claimed rewards might receive lower rewards than those who claimed them later.

Recommendation

We recommend restricting changes in distribution events and calls to `takeClaimingSnapshot` after the first claim.

2.3 Medium

M-1	Incorrect event emitting
Severity	Medium
Status	Fixed in c622b4c3

Description

Here FantiumNFTV3.sol#L386 the event should be `emit Mint(_to, tokenId + i);`.

Recommendation

We recommend changing the event to the `emit Mint(_to, tokenId + i);`.

Client's commentary

> Client: Corrected the mint event as proposed.

M-2	Incorrect input data could break the collection
Severity	Medium
Status	Fixed in c622b4c3

Description

Here FantiumNFTV3.sol#L524-L525 the `_athletePrimarySalesBPS <= 10_000` check should be added. If `_athletePrimarySalesBPS` is > 10_000, then the collection mint would revert because of the underflow.

Recommendation

We recommend adding the following check: `_athletePrimarySalesBPS <= 10_000`.

M-3	Unsafe transfer
Severity	Medium
Status	Fixed in c622b4c3

Description

Some of the tokens do not revert on an unsuccessful transfer. We recommend using the OZ safeTrasnfer method instead of a basic transfer method.
FantiumNFTV3.sol#L409-L413
FantiumNFTV3.sol#L417-L421
FantiumClaimingV1.sol#L591
FantiumClaimingV1.sol#L595

Recommendation

We recommend using the OpenZeppelin safeTransfer library.

M-4	`nextCollectionId` should be limited
Severity	Medium
Status	Fixed in c622b4c3

Description

Due to unnecessary checks on tokenId in claiming, contract `nextCollectionId` should be limited. Otherwise, claiming functionality for tokens of collection with an id >= 100_000 will revert.
FantiumNFTV3.sol#L522

Recommendation

We recommend adding the following check: `nextCollectionId < 1_000_000`.

Client's commentary

> Client: Changed max collection to 1_000_000 and added a require statement to collection setup.

M-5	Collection shares should be limited
Severity	Medium
Status	Fixed in c622b4c3

Description

Shares cannot be > 1e7, otherwise claiming will always revert.
FantiumNFTV3.sol#L547-L553
FantiumNFTV3.sol#L694-L701

Recommendation

We recommend adding the following check:
`_tournamentTokenShare1e7 <= 1e7 && _otherTokenShare1e7 <= 1e7`.

Client's commentary

> Client: Added.

M-6	`maxInvocations` cannot be reduced during an update
Severity	Medium
Status	Fixed in b4acd978

Description

`maxInvocations` cannot be reduced during an update because it will lead to insolvency in the protocol.
FantiumNFTV3.sol#L691

Recommendation

We recommend adding the following check:
`_maxInvocations > collections[_collectionId].maxInvocations`.

Client's commentary

> Client: Please elaborate how this change makes the protocol insolvent.
> As long as maxInvocation > Invocations, there shouldn't be a problem updating the maxInvocation downward in my opinion. Please elaborate
> MB: We recommend at least add check `_maxInvocations >= current_invocations`
> Client: Added

M-7	Modifier `onlyValidCollectionId()` is missing
Severity	Medium
Status	Fixed in c622b4c3

Description

Modifier `onlyValidCollectionId()` should be added here:
FantiumNFTV3.sol#L748-L749

Recommendation

We recommend adding the `onlyValidCollectionId()` modifier.

Client's commentary

> Client: Added the modifier.

M-8	Distribution logic is too flexible
Severity	Medium
Status	Fixed in c622b4c3

Description

Now, the data to calculate the amount for a claim is distributed between the NFT and Claiming contracts. The current architecture may lead to the incorrect distribution of tokens among token holders due to a mistake while updating some collection parameters (e.g. `otherTotalBPS` and `tournamentTotalBPS` should be equal among all collections that are set for distribution; also, there should be a complex check for the collections share to guarantee a fair distribution of rewards among token holders).

Recommendation

We recommend improving the code, so it would still be working according to the desired buiseness logic, but it would minimize the risk of mistakes during the collection parameters update. We recommend implementing the steps below:

1. `tournamentBPS` and `otherBPS` should be moved from the NFT contract to the Claiming contract.
2. Two additional checks should be added while setting up a distribution event:
   Check 1:

```
uint256 a = _tournamentDistributionAmount * 10_000 / tournamentBPS;
uint256 b = a * sum(collection_i_maxInvocations
              * collection_i_tournamentShare / 1e7);
require(b == _tournamentDistributionAmount);
```

Check 2:

```
uint256 a = _otherDistributionAmount * 10_000 / otherBPS;
uint256 b = a * sum(collection_i_maxInvocations
              * collection_i_otherShare / 1e7);
require(b == _otherDistributionAmount);
```

3. The `maxInvocations` parameter cannot be updated during the active distribution for the collection.
4. `collection_i_otherShare` and `collection_i_tournamentShare` should be saved in the distribution event, so these parameters could be seamlessly updated in a collection.

Also, Check 1 and Check 2 should be used in both `updateDistributionEventAmount()` functions in the Claiming contract.

Client's commentary

> Client: This whole logic is replaced. It is now taking the number of tokens minted and multiplies it by the share they hold to determine how much needs to be locked and distributed.

M-9	`fantiumFeeBPS` should be restricted
Severity	Medium
Status	Fixed in c622b4c3

Description

`fantiumFeeBPS` should be <= 10_000, otherwise a claim call will always revert.
FantiumClaimingV1.sol#L242

Recommendation

We recommend adding the following check: `fantiumFeeBPS <= 10_000`.

Client's commentary

> Client: The check is added to the contract.

M-10	An unsafe transfer
Severity	Medium
Status	Fixed in b4acd978

Description

Some of the tokens do not revert on failed `transferFrom`. Also, there are some tokens with a fee on transfer, so the real transferred balance should be checked via `balanceOf()`, otherwise it could lead to funds insolvency.
FantiumClaimingV1.sol#L277-L281

Recommendation

We recommend using the `safeTransfer` library and also adding a check that an exact amount of tokens was transferred to the contract.

Client's commentary

> Client: Changed to using the SafeER20 `safeTransferFrom` Call.
> MB: The same problem as for **HIGH 9**.
> Client: Changed to using the SafeER20 `safeTransferFrom` Call.

M-11	Check for the collection existence is missing
Severity	Medium
Status	Fixed in c622b4c3

Description

There is no check that the collection exists.
FantiumClaimingV1.sol#L315

Recommendation

We recommend adding a check that the collection exists.

Client's commentary

> Client: Added a check for all collectionIDs.

M-12	Possible division by zero
Severity	Medium
Status	Fixed in c622b4c3

Description

There are not enough checks to guarantee there will be no division by zero.
FantiumClaimingV1.sol#L520
FantiumClaimingV1.sol#L527

Recommendation

We recommend adding a check that if `tournamentTotalBPS == 0`, then `tournamentClaim = 0` and the same check for `otherTotalBPS == 0`.

Client's commentary

> Client: Those variables have been removed. Therefore, no chance of them being 0.

M-13	Distribution closing shouldn't be allowed during the active distribution
Severity	Medium
Status	Acknowledged

Description

There are no checks that the distribution ended in the closing function.
FantiumClaimingV1.sol#L624-L631

Recommendation

We recommend adding a check that the distribution can be closed only when `block.timestamp > closeTime`.

Client's commentary

> Client: The team wants to keep the flexibility to close the distributionEvent early if needed. Therefore, the closing function should be able to be called at any time.

M-14	The collection existence check is missing
Severity	Medium
Status	Fixed in c622b4c3

Description

There is no check that the collection exists.
FantiumUserManager.sol#L202

Recommendation

We recommend adding a check that the collection exists.

Client's commentary

> Client: The check is added.

M-15	Variable shadowing
Severity	Medium
Status	Fixed in c622b4c3

Description

There is a `trustedForwarder` variable being set inside the `initialize` function - FantiumNFTV3.sol#L199. The function parameter has the same length as a state variable. In fact, the state variable value doesn't change.

Recommendation

We recommend changing the function parameter name to `_trustedForwarder`.

Client's commentary

> Client: Corrected

M-16	The NFT contract allows multiple ways to avoid KYC
Severity	Medium
Status	Acknowledged

Description

The KYC system for NFT minting does not work. Some easy ways:

1. `FantiumNFTV3.batchMintTo()` checks that `msg.sender` is KYCed but allows specifying `to` who will receive the minted NFT. So, the true NFT receiver is not KYCed.

- FantiumNFTV3.sol#L325-L388

3. NFT is transferable. So, a KYCed account can buy an NFT and transfer it to anyone.

Recommendation

We recommend making necessary changes for the KYC system or removing it.

Client's commentary

> Client: For the team it is no issue if the tokens are at mint transferred to another wallet. The person calling the mint function still needs to be KYC'd. Do you see any other way the KYC can be subverted?
> MB: The issue can be marked as acknowledged.

M-17	Hardcoded values for amounts of ERC20 tokens
Severity	Medium
Status	Fixed in c622b4c3

Description

FantiumClaimingV1.sol#L220, the value of 10 billion payout tokens is hardcoded as $10_000_000_000_000_000, which is equal to 10 billion multiplied by 10^6. If the `payoutToken` is changed in the future with a different `decimals` value, then the contract will need to be redeployed with a new hardcoded value.

Recommendation

We recommend using `ERC20(payoutToken).decimals()` to obtain the decimal multiplier instead of a hardcoded value for more flexibility.

Client's commentary

> Client: Added and reduced to 1B.

M-18	A manager could transfer funds to the Claiming contract instead of the athlete
Severity	Medium
Status	Acknowledged

Description

A manager could transfer funds to the Claiming contract instead of the athlete here FantiumClaimingV1.sol#L303.

Recommendation

We recommend using the athlete's address instead of `_msgSender()`.

Client's commentary

> Client: That is wanted behaviour to allow the Fantium to top up the distributionEvent if needed. Confirmed with the team.

M-19	Claiming could be blocked
Severity	Medium
Status	Fixed in b4acd978

Description

Users that were able to claim before the manager calls this function FantiumClaimingV1.sol#L316-L337 will have to wait until the athlete transfers tokens.

Recommendation

We recommend calling `addDistributionAmount` in the `updateDistribtionTotalEarningsAmounts()` and `updateDistributionEventCollectionIds` functions.

Client's commentary

> Client: The updates will be made by the platform manager, not by the athlete. We configure the contracts for the athletes. Therefore, we cannot call addDistributionAmount for them. Also, because of 18.)
> Therefore, we need to stop the claim process to make sure the full amount is paid in to ensure no overclaiming can happen.

M-20	The check can be manipulated
Severity	Medium
Status	Fixed in b4acd978

Description

A platform manager can by mistake update shares in the NFT contract and call `takeClaimingSnapshot` which will lead to the situation where the athlete should add more funds to the distribution before closing it FantiumClaimingV1.sol#L754.

Recommendation

We recommend changing the `closeDistribution` function.

Client's commentary

> Client: Removed the mentioned check to let closing be independent of variable changes.

M-21	Incorrect work with decimals
Severity	Medium
Status	Fixed in b4acd978

Description

`DistributionEvent.totalTournamentEarnings` and `DistributionEvent.totalOtherEarnings` are stored for the current token's decimals (FantiumClaimingV1.sol#L63). If the new token is set with an other number of decimals after the distribution is created, these variables will contain incorrect values. Therefore, `DistributionEvent.tournamentDistributionAmount` and `DistributionEvent.otherDistributionAmount` will contain incorrect values as well (FantiumClaimingV1.sol#L698). This will lead to an incorrect amount of distributed funds.

Recommendation

We recommend storing `totalTournamentEarnings` and `totalOtherEarnings` without decimals. In such cases `amountPaidIn`, `tournamentDistributionAmount`, and `otherDistributionAmount` should be stored without decimals too.

Client's commentary

> Client: As we have moved the payout calculation per token forward to the snapshot. The amounts need to be stored with decimals to capture the decimals of the resulting claimAmount.
> I have stored the payout token on the distributionEvent. Even if the payOutToken is changed on the contract, the distributionEvent will remain with the same "old payoutToken" or must be closed.
> Managers who are entering the quantities will be aware to always use amounts with decimals when entered.

M-22	Unnecessary check reverts function call.
Severity	Medium
Status	Fixed in 0ad51f89

Description

It's impossible to call `updateDistributionEventCollectionIds()` or `updateDistribtionTotalEarningsAmounts()` if `tournamentDistributionAmount`+`otherDistributionAmount` stays the same. `topUpDistributionEvent()` reverts if it's not needed to transfer additional funds from the athlete's address.
FantiumClaimingV1.sol#L536
So, `updateDistributionEventCollectionIds()` and `updateDistribtionTotalEarningsAmounts()` revert too in such cases. But the manager may want to change these parameters while the total distribution amount stays the same. For example, they want to change the collections' order, from one collection to another with identical parameters. Or they want to set another ratio between tournamentEarnings and otherEarnings.

Recommendation

We recommend removing a check `payInAmount > 0` from `topUpDistributionEvent()`.

M-23	Anyone can block the payout token by changing its amount by 1 wei
Severity	Medium
Status	Fixed in 0ad51f89

Description

There is a check in `updatePayoutToken()` that the balance should be 0.
FantiumClaimingV1.sol#L749
So, anyone can send 1 wei to the balance and make it impossible to change the payout token.

Recommendation

We recommend removing a zero balance check from `updatePayoutToken()`. Now it's not required because the token address is stored in the snapshot.

Client's commentary

> Client: Adjusted by removing the check.

M-24	A missed requirement
Severity	Medium
Status	Fixed in 0ad51f89

Description

Although there is a check in the `takeClaimingSnapshot` function (FantiumClaimingV1.sol#L686), it is more crucial to include it in the `updateDistributionEventCollectionIds()` function (FantiumClaimingV1.sol#L354). Implementing this check makes the previous check unnecessary, as it becomes impossible to set an empty collectionIds array for the distribution event.

Recommendation

We recommend adding the check `distributionEvents[_distributionEventId].collectionIds.length > 0` to the `updateDistributionEventCollectionIds()` function.

Client's commentary

> Client: Added.

M-25	`takeClaimingSnapshot()` call will lead to an inconsistent state
Severity	Medium
Status	Acknowledged

Description

This function can block the ongoing distribution event until the funds are paid or the event parameters are returned back.
FantiumClaimingV1.sol#L689

Recommendation

We recommend removing this function or adding a `topUpDistributionEvent()` call at the end of the function.

Client's commentary

> Client: The top-up might fail due to missing the ERC20 approval from the athlete that `topUpDistribtionEvent` would need. Then the whole snapshot fails. In our perspective it is better to have the snapshot taken, the claiming halted and needs to be topped up by the athlete to continue payout.
> Reason being also that the snapshot should be at a certain date and time. Athlete topup and snapshot therefore decoupled. Lastly, we could ask the athlete to already approve USDC spending in advance, but that is also not really a good practise to just approve USDC spending without a direct following payment and keeping an open allowance IMO.
> MB: This function can cause an unfair distribution of tokens among NFT holders if the manager calls it after earning share was changed for the specific collection, but this can be acknowledged without any severe impact on the security.

2.4 Low

L-1	Possible incorrect event emitting
Severity	Low
Status	Fixed in 0ad51f89

Description

There are no checks whether an address was previously kyced or not, so it is possible that events will be emited > 1.
FantiumNFTV3.sol#L229
FantiumNFTV3.sol#L240

Recommendation

We recommend adding a check that a user doesn't have kyc or was already removed from kyc.

Client's commentary

> Client: Added to the new KYCManager Contract. We have removed the previous logic of KYClisting and allowlisting that was sitting on the NFT contract and have now the NFT contract listening to the KYC manager contract for the state of KYC and allowlist.
> MB: There is still a problem with adding/removing the IDENT status and allowed contracts in the KYCManager contract.
> Client: Adjusted.
> MB: There is still a problem with adding/removing allowed contracts.
> Client: Added a check to only trigger the change on the allowed contract if it had the opposite previous state.

L-2	A function can be called with `amount == 0`
Severity	Low
Status	Fixed in c622b4c3

Description

A function can be called with `amount == 0`.
FantiumNFTV3.sol#L325-L329

Recommendation

We recommend adding an `amount > 0` check.

Client's commentary

> Client: Added.

L-3	Minting insolvency
Severity	Low
Status	Fixed in c622b4c3

Description

It is better to use require, because some integrations could break with the current logic.
FantiumNFTV3.sol#L331

Recommendation

We recommend using require to restrict the function calling with `amount > 10`.

Client's commentary

> Client: Added.

L-4	Unnecessary checks
Severity	Low
Status	Fixed in c622b4c3

Description

There are some checks that are unnecessary, so they can be removed.
FantiumNFTV3.sol#L349-L353

Recommendation

We recommend removing these checks.

Client's commentary

> Client: Removed.

L-5	A zero address check
Severity	Low
Status	Fixed in b4acd978

Description

Some tokens could be lost because the address is set to zero.
FantiumNFTV3.sol#L543
FantiumClaimingV1.sol#L241
FantiumUserManager.sol#L79-L80

Recommendation

We recommend adding a check that the address is not zero.

Client's commentary

> Client: Adjusted
> MB: There are still no checks for zero addresses in the `FantiumUserManager` contract.

L-6	A front-run risk
Severity	Low
Status	Acknowledged

Description

It is possible to buy NFTs cheaper before the price update.
FantiumNFTV3.sol#L692

Recommendation

We recommend using a private transactions pool for the price updating.

Client's commentary

> Client: Acknowledged.

L-7	An unchecked timestamp
Severity	Low
Status	Acknowledged

Description

There are not enough timestamp checks.
FantiumNFTV3.sol#L719
FantiumNFTV3.sol#L536

Recommendation

We recommend adding checks to make sure that a new timestamp cannot be less than the current timestamp (block.timestamp).

Client's commentary

> Clent: We want to keep the option to have the timestamp in the past if the athlete wants to launch right away.

L-8	Events missing
Severity	Low
Status	Fixed in c622b4c3

Description

There are missing events for the important variables update.
FantiumNFTV3.sol#L826
FantiumNFTV3.sol#L841
FantiumNFTV3.sol#L977
FantiumClaimingV1.sol#L607
FantiumClaimingV1.sol#L614
FantiumClaimingV1.sol#L621
FantiumClaimingV1.sol#L647-L651

Recommendation

We recommend adding events emitting on these parameters update.

L-9	Bad UX on claiming
Severity	Low
Status	Acknowledged

Description

After claiming a user will lose all approvals on their tokens which can be inconvenient.
FantiumNFTV3.sol#L803

Recommendation

We recommend adding information for users that they will lose approvals on their tokens after claiming.

L-10	An inconvenient token update
Severity	Low
Status	Acknowledged

Description

After the token update each collection price should be updated.
FantiumNFTV3.sol#L841

Recommendation

We recommend using only tokens with the same price.

L-11	Initialization front-running
Severity	Low
Status	Acknowledged

Description

It is possible to front-run initialization of the Claiming contract.
FantiumClaimingV1.sol#L135-L150

Recommendation

We recommend checking that the Claiming contract was initialized correctly before setting its address to the NFT contract.

Client's commentary

> Client: Acknowledged.

L-12	No `forwarder` address check
Severity	Low
Status	Acknowledged

Description

There are no checks on the `forwarder` address in the initialize function.
FantiumClaimingV1.sol#L160-L162

Recommendation

We recommend adding a check for the `forwarder` address.

Client's commentary

> Client: I have removed the forwarder from the constructor, integrated the ERC2771 directly and have a setter function for the forwarder now.

L-13	Bad work with decimals
Severity	Low
Status	Fixed in c622b4c3

Description

6 decimals are hardcoded in the Claiming contract which can lead to incorrect work with token amounts in the future.
FantiumClaimingV1.sol#L218-L220

Recommendation

We recommend adjusting work with decimals to make it more flexible.

L-14	An unnecessary parameter
Severity	Low
Status	Fixed in c622b4c3

Description

`_amountWithDecimals` is unnecessary in the `addDistributionAmount()` function.
FantiumClaimingV1.sol#L262-L269

Recommendation

We recommend using `distributionEvents[_distributionEventId].tournamentDistributionAmount + distributionEvents[_distributionEventId].otherDistributionAmount` instead.

L-15	Unused logic
Severity	Low
Status	Acknowledged

Description

There is a lot of unused logic in the Manager contract.
FantiumUserManager.sol#L95-L140
FantiumUserManager.sol#L185-L242

Recommendation

We recommend removing unused logic.

Client's commentary

> Client: It is now being used by KYC and allowlist users on the NFT contract. Logic newly integrated.

L-16	Unchecked range for time
Severity	Low
Status	Acknowledged

Description

There are not enough checks for `_startTime` and `_closeTime` in the Claiming contract.
FantiumClaimingV1.sol#L208-L211
FantiumClaimingV1.sol#L352-L353

Recommendation

We recommend adding a check that `_startTime > block.timestamp`, also a new `_startTime` cannot be less than the previous `_startTime`.

Client's commentary

> Client: We want to keep the option open that the _startTime is in the past so people can claim immediately. ClosingTime should be in the future. The check is added.

L-17	Unnecessary checks
Severity	Low
Status	Fixed in c622b4c3

Description

There are several unnecessary checks.
FantiumClaimingV1.sol#L408-L412
FantiumClaimingV1.sol#L636-L639

Recommendation

We recommend removing unnecessary checks.

Client's commentary

> Client: Duplicate checks removed.

L-18	Not checking the length on `_addresses` and `_increaseAllocations`
Severity	Low
Status	Fixed in c622b4c3

Description

At line FantiumNFTV3.sol#L262 the `batchAllowlist` function is defined. It takes two arrays as an argument and then loops over them. It is expected that two arrays would have the same length as they contain corresponding data. But their lengths aren't compared.

Recommendation

We recommend comparing that the arrays have the same lengths.

Client's commentary

> Client: The check is added.

L-19	Unnecessary casting to `uint256`
Severity	Low
Status	Fixed in c622b4c3

Description

Unnecessary casts were spotted here - FantiumClaimingV1.sol#L580 and here - FantiumNFTV3.sol#L462.

Recommendation

It's not recommended to cast to `uint256` there, as arguments have the `uint256` types themselves.

Client's commentary

> Client: Casting removed.

L-20	`onlyManager()` and `onlyRole(PLATFORM_MANAGER_ROLE)` have the same logic
Severity	Low
Status	Fixed in c622b4c3

Description

There are some places where the `onlyManager()` modifier is used - FantiumClaimingV1.sol#L201 and `onlyRole(PLATFORM_MANAGER_ROLE)` - FantiumClaimingV1.sol#L298. Those modifiers have the same logic. There is no need to use both of them.

Recommendation

We recommend coming to a one type of modifier when checking that the user is the protocol manager.

Client's commentary

> Client: onlyManager removed.

L-21	Equation can be simplified
Severity	Low
Status	Fixed in c622b4c3

Description

The calculation of the token collection can be simplified.
TokenVersionUtil.sol#L21-L22

Recommendation

We recommend changing the calculation to `uint256 collectionOfToken = _tokenId /ONE_MILLION;`.

Client's commentary

> Client: Simplified to this calculation.

L-22	Bad name for the function
Severity	Low
Status	Fixed in c622b4c3

Description

Th function name was copypasted from the previous function.
FantiumClaimingV1.sol#L306-L309
FantiumClaimingV1.sol#L338-L342

Recommendation

We recommend changing the function name.

Client's commentary

> Client: Already adjusted!

L-23	A function can be optimized
Severity	Low
Status	Fixed in c622b4c3

Description

The function can be optimized in two places:
FantiumClaimingV1.sol#L548-L560
FantiumClaimingV1.sol#L553-L558.

Recommendation

We recommend adding `if(!tokenNrClaimed)` before the loop and `break` inside `if` in the loop.

Client's commentary

> Client: Optimization added.

L-24	Bad pause management
Severity	Low
Status	Fixed in c622b4c3

Description

Pause management is very inconvenient right now, because all manager functions pause with the user functions which blocks the manager from making changes in the contract.

Recommendation

We recommend rebuilding pause management and giving to the manager the ability to change some important parameters when the contract is paused.

Client's commentary

> Client: Removed the "whenNotPaused" modifier on the NFT, UserManager and claiming contract on several update functions to allow the manager to update contract state in the case of paused state.

L-25	A limit of the distribution amount as one billion is not correct
Severity	Low
Status	Fixed in b4acd978

Description

FantiumClaimingNFTV1 has these lines:

- FantiumClaimingV1.sol#L216-L222.
  The comments state that it should "check if the amount is less than a billion".
  By the way, it compares with number `10_000_000_000_000_000`, which is in fact ten billion with 6 decimals.

Recommendation

We recommend updating either the comparison or the comment to adjust to true designed limitations.

Client's commentary

> Client: Adjusted the limit to fit the comment.
> MB: The revert message is still incorrect.
> Client: Revert message adjusted.

L-26	Version calculation can be simplified
Severity	Low
Status	Fixed in b4acd978

Description

Token's version calculation can be simplified. Now it's calculated as `((_tokenId % ONE_MILLION) - (_tokenId % TEN_THOUSAND)) / TEN_THOUSAND` (TokenVersionUtil.sol#L22). But it's equal to `(_tokenId % ONE_MILLION) / TEN_THOUSAND`.

Recommendation

We recommend simplifying the version calculation.

Client's commentary

> Client: Adjusted.

L-27	An unnecessary internal function onlyRole modifier
Severity	Low
Status	Fixed in b4acd978

Description

The internal function `FantiumClaimingV1.triggerClaimingSnapshot()` has a `onlyRole(PLATFORM_MANAGER_ROLE)` modifier (FantiumClaimingV1.sol#L665). This function can't be called from outside, so any role modifier is unnecessary. Also, every function calling `triggerClaimingSnapshot()` already has this modifier.

Recommendation

We recommend removing this modifier.

Client's commentary

> Client: Check removed.

L-28	`payOutToken` and `payoutToken` variables differ only in one symbol.
Severity	Low
Status	Fixed in 0ad51f89

Description

It's easy to make a mistake using variables `payOutToken` and `payoutToken`. It's better to use another name for the local variable.

Recommendation

We recommend using another name for `payOutToken`.

Client's commentary

> Client: Renamed to globalPayoutToken.

L-29	Functions `topUpDistributionEvent()` and `addDistributionAmount()` have duplicated functionality.
Severity	Low
Status	Fixed in 0ad51f89

Description

Functions `topUpDistributionEvent()` and `addDistributionAmount()` have duplicated functionality.

Recommendation

We recommend taking out the logic to one function.

Client's commentary

> Client: That is clear. The major difference is that addDisbtrutionAmount can be called by Fantium platformManager as well and the funds can be paid in by the calling party. We want that option in case Fantium does the claiming initial payIn for the athlete.
> Once we change something about it, it will be automatically topped up from the athlete. We can then reimburse them if needed.

L-30	The `maxInvocations` parameter check
Severity	Low
Status	Acknowledged

Description

The FantiumNFTV3 contract allows to set off the `maxInvocations` parameter for collection and change its value in the future. Current checks allow not more than 9999 tokens (`maxInvocations < 10000`) (Max token id inside one token version would be 9998).

Recommendation

We recommend using a non-strict (<=) check instead of a strict one (<).

Client's commentary

> Client: Acknowledged. But in case we want to alter the tokenIds to start with X000001 at some point, that would be something to easily overlook and break the contract. Would not adjust for now.

L-31	An unnecessary check
Severity	Low
Status	Fixed in 0ad51f89

Description

There is an unnecessary check inside the `claim` function. It compares `closeTime` with a 0 value despite the fact that such value cannot ever be set to that parameter.

Recommendation

We recommend removing that check.

Client's commentary

> Client: Added.

3. ABOUT MIXBYTES

MixBytes is a team of blockchain developers, auditors and analysts keen on decentralized systems. We build opensource solutions, smart contracts and blockchain protocols, perform security audits, work on benchmarking and software testing solutions, do research and tech consultancy.

Contacts

https://github.com/mixbytes/audits_public

https://mixbytes.io/

hello@mixbytes.io

https://twitter.com/mixbytes