Unblock Labs

An EatTheBlocks Company

Audit report

BandZai

March 2023

Table of Contents

Summary

This report has been prepared by Unblock Labs for BandZai to discover issues and vulnerabilities in the source code of their play to earn smart contracts as well as any contract dependencies used in the project. A comprehensive examination has been performed utilising Static Analysis and Manual Code Review techniques

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards. Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project summary

Project name	**BandZai**
Platform	Polygon
Language	Solidity
Project url	https://www.bandzai.games
Codebase	https://github.com/BandZaiGame/BandZaiGame
Revised Codebase	Commit 8b571780c9389a7a6f16d39a3069473332b4ed0e

Audit summary

Delivery date	March 14, 2023
Methodology	Static Analysis, Manual Review

Vulnerability summary

Level	**Total**	**Acknowledge**	**Mitigated**	**Resolved**
🔴 Critical	0	0	0	0
🟠 High	33	3	1	29
🟠 Medium	64	4	0	60
🟡 Low	97	3	0	94
🔵 Information	12	3	0	9

Level	Total	Acknowledge	Mitigated	Resolved
🟢 Discussion	0	0	0	0

Audit scope

ID	Contract	SHA256 checksum
ALC	AlchemyV1.sol	404af6ff45555e09f7546627297ae3ed5e2f8542
BZA	BandZaiAddresses.sol	742a2182d13ad1554306d4c1a78c946216c94b62
BZT	BandZaiToken.sol	9c06bfe2a938689aa8e64a8129c488fcef79ae07
CHK	ChickenNFT.sol	182ef3226c6663b70a96b9d1683912e6f80fbcd8
CLB	ClaimBzai.sol	8a32fe93da50d051ba92a111a2931c61d8ce4ede
CLN	ClaimNFT.sol	7057022e1622a942fb9a03e0a17fbdbd4d641443
CLT	ClaimTeamAndMarketingBzai.sol	bf847c6b9c2d4516da8ab8748fabda74402c6c68
DWR	DailyWeeklyRanking.sol	ab2794d930f83d93323c64efa6ae8de910a93149
DMZ	DelegateMyZai.sol	f2cecf50f09cfef278b02678d6f297a9cce1c3aa
EGG	EggsNFT.sol	0be85150de30baac8a588150dafa30ebdd197cb1
GLD	GuildeDelegation.sol	e2ba6744fad95cba2a6b21dfd9a2336cdad768e6
INT	Interfaces.sol	c7185a080737bbd174b44157f9c063a0e7185d6e
IPF	IpfsIdStorage.sol	3a0fd70a083b875ecd39eabdcaf9e7b0ebd22b17
LAM	LaboManagement.sol	80d74d590326c4d3c88d3cb96141ea92748afc5f
LAB	Laboratory.sol	cbea36c944a7a6878584bbd5153e2b040d996178
LVL	LevelStorage.sol	69110b69800bd0dc9d4b0b7c00489af3c56c1f05
LIQ	LiquidityMining.sol	2a4a6b627202b65da8eb1b520998e8ff683d0018
LOT	LootProgress.sol	d74833e181ac7ea94f41057ed0a3abfbf7d1fa0d
MKP	MarketPlace.sol	c0c9a3260a668ae9c95628a03924490b85eb31ab
MKZ	MarketZai.sol	90e1a1282bb99cfbd361af378afab8a4e7af69ed
SIG	MultiSigWallet.sol	03b443f53548b9727747253b96df1cfa4d213e35
NRS	Nursery.sol	d1f4d6bd48971aad435f3f7a9a88ea0095eff26b
NRM	NurseryManagement.sol	06f1a97c42c8a85b04929719d1867d289a6c18c6
OCC	OpenAndCloseCenter.sol	a86dd7a1e75b909e4276b7acabb2edb69b87cfc8

ID	Contract	SHA256 checksum
ORA	Oracle1.sol	6d7ea735cd4536a68bcc457252f4e611374b3d2f, ffad6b8de2f9a5aec04cfe09d226faa1d1adff97
PAY	Payments.sol	5df5bb72ccb60de396f74f1626b6a4dfefe6d10b
POT	PotionNFT.sol	46dbb2704cf0ff9dd488fb4dfb85f00ff7e49748
RCR	ReserveForChallengeRewards.sol	55e525da457b9ef7dfc9913f217bba8c85941701
RWR	ReserveForWinRewards.sol	1f9b88a3b10259d9a2891c7901f2644e4d122d24
RPP	RewardsPvP.sol	525a31d601be593c9727c66795cb5a79593ce9c5
RRF	RewardsRankingFound.sol	0780a730eaab2e2dbf65cbff9810124a946582ad
RWT	RewardsTournament.sol	8231f1d5494682ee5e454b0b6cb30ed123a1f84d
RWF	RewardsWinningFound.sol	f77ec5175f141b449ae5e61850a844346bc1427d
TRC	TrainingCenter.sol	8ca1b38f884a5d0cb4c92eb9aa7556989aaa3b4d
TRM	TrainingManagement.sol	9105a6dcd081e5bd53ae0a347f0e10d2e0666fce
ZFT	ZaiFighting.sol	f84e56a2ac88c8e4851666e61fdc8400a6f2b922
ZFL	ZaiFightingLibrary.sol	4735cc53ebeca13288480065bb7ecb15f8e9173f
ZMT	ZaiMeta.sol	931fc0a4a129a908979048b6fe754f72d7afb2d8
ZNF	ZaiNFT.sol	4e7a1ef73099d5ee669f1174e710b8826377e5e5
ZST	ZaiStats.sol	6bb50b2d6cac4fa22410a94dbd3670d5fa6aa1f8
GLB	Global	

Findings

ID	Title	Category	Severity	Status
ALC-01	Invalid power attribution	Volatile Code	● High	Resolved
BZA-01	No events emitted	Language Specific	● High	Resolved
BZA-02	Addresses upgradability	Coding Style	● High	Resolved
CHK-01	Check Effects Interactions pattern violation	Volatile Code	● High	Resolved

ID	Title	Category	Severity	Status
CLN-01	Check Effects Interactions pattern Violation	Volatile Code	● High	Resolved
CLN-02	Unbounded loop in claimAllNFTs	Volatile Code	● High	Resolved
EGG-01	Check Effects Interactions pattern Violation	Volatile Code	● High	Resolved
GLB-01	Centralization related risks	Centralization / Privilege	● High	Acknowledge
GLB-02	Anti bot prevention	Coding Style	● High	Mitigated
IPF-01	EnumerableSet does not guarantee order	Volatile Code	● High	Resolved
LAB-01	Missing onlyAuth modifier in updateCreditLastUpdate	Volatile Code	● High	Resolved
LAM-01	EnumerableSet does not guarantee order	Volatile Code	● High	Resolved
LIQ-01	Invalid variable decrement	Logical Issue	● High	Resolved
MKP-01	bidForNft() does not check for revenues	Logical Issue	● High	Resolved
MKZ-01	_calculateDutchPrice reduction does not work within a day	Logical Issue	● High	Resolved
NRM-01	Invalid maturity date	Logical Issue	● High	Resolved
NRS-01	Check Effects Interactions pattern Violation	Volatile Code	● High	Resolved
OCC-01	LP token can be changed and prevent users from withdrawing	Volatile Code	● High	Resolved
OCC-02	Check Effects Interactions pattern Violation	Volatile Code	● High	Resolved
ORA-01	Weak sources of randomness	Volatile Code	● High	Acknowledge
RCR-01	Mix of responsibility	Volatile Code	● High	Resolved
RRF-01	Mix of responsibility	Volatile Code	● High	Resolved

ID	Title	Category	Severity	Status
RWR-01	Mix of responsibility	Volatile Code	● High	Resolved
SIG-01	Upgradability of signers and required confirmations	Volatile Code	● High	Acknowledge
TRC-01	Burn without approval of the owner	Centralization / Privilege	● High	Resolved
TRC-02	Check Effects Interactions pattern Violation	Volatile Code	● High	Resolved
TRM-01	Missing input validation	Volatile Code	● High	Resolved
TRM-02	Potential underflow in _updateZai	Volatile Code	● High	Resolved
TRM-03	Check Effects Interactions pattern Violation	Volatile Code	● High	Resolved
ZFL-01	Mix of index and power values	Volatile Code	● High	Resolved
ZFT-01	Missing validation of reward repartition	Volatile Code	● High	Resolved
ZNF-01	Potential loss of "piggybank"	Volatile Code	● High	Resolved
ZST-01	Invalid event emitted	Volatile Code	● High	Resolved
ALC-02	Duplicate test in useAlchemy	Volatile Code	● Medium	Resolved
BZA-03	Random number Oracle complexity	Language Specific	● Medium	Resolved
CHK-02	Duplicate functionality from base class	Gas Optimization	● Medium	Resolved
CLB-01	No events emitted	Language Specific	● Medium	Resolved
CLB-02	resetNFT() does not clean data structures	Gas Optimization	● Medium	Resolved
CLN-03	No events emitted	Language Specific	● Medium	Resolved
CLT-01	Previous assignation of tokens not take into account	Logical Issue	● Medium	Resolved
CLT-02	No events emitted	Language Specific	● Medium	Resolved

ID	Title	Category	Severity	Status
DMZ-01	Function canUseZai() does not check expired scholarships	Volatile Code	● Medium	Resolved
DMZ-02	ZaiAddress should be cached and immutable	Coding Style	● Medium	Resolved
DWR-01	Missing input validation	Volatile Code	● Medium	Resolved
EGG-02	No events emitted	Language Specific	● Medium	Resolved
GLB-03	Use of ERC721Enumerable	Gas Optimization	● Medium	Resolved
GLD-01	delegateNFTs should use safeTransferFrom and implement onErc721Received	Volatile Code	● Medium	Resolved
GLD-02	Possible underflow exception in delegateNFTs()	Volatile Code	● Medium	Resolved
GLD-03	Unbounded loop in _getScholarNFTs and _getGuildNFTs	Gas Optimization	● Medium	Resolved
LAB-02	No restriction on _preMintNumber in the constructor	Volatile Code	● Medium	Resolved
LAB-03	Addresses from gameAddresses should be cached	Gas Optimization	● Medium	Resolved
LAM-02	Random revert in workInASpot()	Volatile Code	● Medium	Resolved
LAM-03	Missing input validation	Volatile Code	● Medium	Resolved
LAM-04	No events emitted	Language Specific	● Medium	Resolved
LAM-05	PotionSold event should expose potionId and buyer address	Coding Style	● Medium	Resolved
LAM-06	Unbounded loop in getUnsoldPotions()	Gas Optimization	● Medium	Resolved
LIQ-02	getMiningStarted returns true when not started	Volatile Code	● Medium	Resolved

ID	Title	Category	Severity	Status
LIQ-03	No events emitted	Coding Style	● Medium	Resolved
LOT-01	Addresses from gameAddresses should be cached	Gas Optimization	● Medium	Resolved
LVL-01	Missing validation in getRandomZaiFromLevel	Volatile Code	● Medium	Acknowledge
LVL-02	No events emitted	Coding Style	● Medium	Resolved
MKP-02	Approval can be restricted to the token only in sellNft()	Centralization / Privilege	● Medium	Resolved
MKP-03	NFT can be listed multiple times in sellNft()	Volatile Code	● Medium	Resolved
MKZ-02	No events emitted	Coding Style	● Medium	Resolved
MKZ-03	Addresses from gameAddresses should be cached	Gas Optimization	● Medium	Resolved
NRM-02	No events emitted	Language Specific	● Medium	Resolved
NRM-03	Mix of responsibility with base class	Coding Style	● Medium	Resolved
NRS-02	Pre-minting should not be done in the constructor of the contract	Volatile Code	● Medium	Acknowledge
OCC-03	No events emitted	Language Specific	● Medium	Resolved
OCC-04	housesStates should use an enum	Gas Optimization	● Medium	Resolved
ORA-02	Parameter _id does not add randomness	Gas Optimization	● Medium	Resolved
PAY-01	No events emitted	Coding Style	● Medium	Resolved
POT-01	Burn token without approval	Centralization / Privilege	● Medium	Resolved
POT-02	No events emitted	Coding Style	● Medium	Resolved
RPP-01	No events emitted	Coding Style	● Medium	Resolved

ID	Title	Category	Severity	Status
RRF-02	setGameAddresses() should call updateAddresses()	Coding Style	● Medium	Acknowledge
RRF-03	No events emitted	Coding Style	● Medium	Resolved
RWF-01	setHourlyBlockQuantity can change reward emission	Centralization / Privilege	● Medium	Resolved
RWF-02	No events emitted	Coding Style	● Medium	Resolved
RWF-03	Low resolution for bonusMult	Coding Style	● Medium	Resolved
RWT-01	No events emitted	Coding Style	● Medium	Resolved
TRC-03	No restriction on _preMint in the constructor	Volatile Code	● Medium	Acknowledge
TRC-04	No events emitted	Coding Style	● Medium	Resolved
TRM-04	Differences between cleanSpot and kickCoachFromSpot	Volatile Code	● Medium	Resolved
TRM-05	No events emitted	Coding Style	● Medium	Resolved
TRM-06	slotStatus should be an enum	Gas Optimization	● Medium	Resolved
ZFL-02	Gas optimisation in updateFightingProgress()	Gas Optimization	● Medium	Resolved
ZFT-02	Rounding error in _paySchoolarAndOwner()	Volatile Code	● Medium	Resolved
ZFT-03	No events emitted	Coding Style	● Medium	Resolved
ZMT-01	Missing address(0) validation	Volatile Code	● Medium	Resolved
ZMT-02	No events emitted	Coding Style	● Medium	Resolved
ZMT-03	Gas optimisation in _createZaiDatas()	Gas Optimization	● Medium	Resolved
ZNF-02	Missing address(0) validation	Volatile Code	● Medium	Resolved
ZNF-03	Duplicate functionality from base class	Gas Optimization	● Medium	Resolved
ZST-02	Events emitted before state change	Coding Style	● Medium	Resolved
ZST-03	Unused properties on-chain	Gas Optimization	● Medium	Resolved

ID	Title	Category	Severity	Status
ZST-04	Gas optimisation in updateCounterWinLoss()	Gas Optimization	● Medium	Resolved
ALC-03	Gas optimisation in useAlchemy	Gas Optimization	● Low	Resolved
ALC-04	Cache addresses from gameAddresses	Gas Optimization	● Low	Resolved
BZA-04	Unused variable	Gas Optimization	● Low	Resolved
BZT-01	Ownership of BandZaiToken contract	Centralization / Privilege	● Low	Resolved
CLB-03	Gas optimisation in setAdvisorsVesting()	Gas Optimization	● Low	Resolved
CLN-04	Missing input validation	Language Specific	● Low	Resolved
CLT-03	Potential overflow in setMarketingVesting	Volatile Code	● Low	Resolved
CLT-04	Check Effects Interactions pattern Violation	Volatile Code	● Low	Resolved
CLT-05	Redundant code	Gas Optimization	● Low	Resolved
CLT-06	Gas optimisation in setTeamVesting and setMarketingVesting	Gas Optimization	● Low	Resolved
DMZ-03	Pagination should be done by the caller	Coding Style	● Low	Resolved
DWR-02	Duplicate variable	Gas Optimization	● Low	Resolved
DWR-03	Gas optimisation in _updateDailyRanking()	Gas Optimization	● Low	Resolved
DWR-04	Use bytes to compare strings	Gas Optimization	● Low	Resolved
EGG-03	updateMaturity is never called	Volatile Code	● Low	Resolved
EGG-04	Missing owner validation in coverEggWithChicken	Volatile Code	● Low	Resolved
EGG-05	Missing input validation	Volatile Code	● Low	Resolved
EGG-06	Shadows of existing variable name	Language Specific	● Low	Resolved

ID	Title	Category	Severity	Status
EGG-07	No added value in _isCover	Gas Optimization	🟡 Low	Resolved
EGG-08	Gas optimisation in coverEggWithChicken	Gas Optimization	🟡 Low	Resolved
GLD-04	Unused ERC721Holder import	Volatile Code	🟡 Low	Resolved
GLD-05	No added value in _getRentingDatas	Gas Optimization	🟡 Low	Resolved
INT-01	Structure optimisation	Gas Optimization	🟡 Low	Resolved
IPF-02	No added value in _getIdsLength	Gas Optimization	🟡 Low	Resolved
LAB-04	Duplicate functionality	Gas Optimization	🟡 Low	Resolved
LAM-07	Potions from burnt Laboratory are still listed for sale	Volatile Code	🟡 Low	Resolved
LAM-08	PotionSold event should be emitted after the transfer	Coding Style	🟡 Low	Resolved
LAM-09	Addresses from gameAddresses should be cached	Gas Optimization	🟡 Low	Resolved
LAM-10	Unused property	Gas Optimization	🟡 Low	Resolved
LAM-11	setWorkingSpotPrice should be external	Gas Optimization	🟡 Low	Resolved
LAM-12	Structure/Mapping optimisations	Gas Optimization	🟡 Low	Resolved
LAM-13	createdPotionsForLab can be recreated offchain from events	Gas Optimization	🟡 Low	Resolved
LIQ-04	Missing input validation	Volatile Code	🟡 Low	Resolved
LIQ-05	ReentrancyGuard not used	Gas Optimization	🟡 Low	Resolved
LIQ-06	Gas optimisation in updatePool()	Gas Optimization	🟡 Low	Resolved
LIQ-07	Gas optimisation in claim()	Gas Optimization	🟡 Low	Resolved

ID	Title	Category	Severity	Status
LIQ-08	BZAIPerBlock should be constant	Gas Optimization	🟡 Low	Resolved
LOT-02	Use of string parameter in event NewLootResult	Gas Optimization	🟡 Low	Resolved
LOT-03	Loop can be replaced by a division	Gas Optimization	🟡 Low	Resolved
MKP-04	User can bid on his own NFT in bidForNft()	Volatile Code	🟡 Low	Resolved
MKP-05	Check Effects Interactions pattern Violation in buyNft()	Volatile Code	🟡 Low	Resolved
MKP-06	Check Effects Interactions pattern Violation in acceptBid()	Volatile Code	🟡 Low	Resolved
MKP-07	offerDuration is initialised from non-constant variable	Language Specific	🟡 Low	Resolved
MKP-08	Unnecessarily import of EnumerableSet.sol	Gas Optimization	🟡 Low	Resolved
MKP-09	BZAI property should be immutable	Gas Optimization	🟡 Low	Resolved
MKP-10	_myProposals can be recreated off-chain	Gas Optimization	🟡 Low	Resolved
MKZ-04	duplicate function "setBlockPerday" and "setBlockPerDay"	Gas Optimization	🟡 Low	Resolved
MKZ-05	BZAI property should be immutable	Gas Optimization	🟡 Low	Resolved
MKZ-06	Gas optimisation in _randomMint() and _getZaiPrice()	Gas Optimization	🟡 Low	Resolved
NRM-04	Unused property	Gas Optimization	🟡 Low	Resolved
NRM-05	Mapping can be in refactored in struct	Gas Optimization	🟡 Low	Resolved
NRS-03	Unused variables	Coding Style	🟡 Low	Resolved

ID	Title	Category	Severity	Status
NRS-04	Unused property	Gas Optimization	🟡 Low	Resolved
NRS-05	Unused function	Gas Optimization	🟡 Low	Resolved
OCC-05	Mapping can be in refactored in struct	Gas Optimization	🟡 Low	Resolved
OCC-06	Addresses from gameAddresses should be cached	Gas Optimization	🟡 Low	Resolved
ORA-03	No functionality added between Oracle1 and Oracle2	Coding Style	🟡 Low	Resolved
PAY-02	Owner and DAO address is the same	Centralization / Privilege	🟡 Low	Resolved
PAY-03	block.timestamp not required in events	Gas Optimization	🟡 Low	Resolved
PAY-04	BZAI property should be immutable	Gas Optimization	🟡 Low	Resolved
POT-03	Missing input validation	Volatile Code	🟡 Low	Resolved
POT-04	Gas optimisation in offerPotion() and mintPotionForSale()	Gas Optimization	🟡 Low	Resolved
RCR-02	Gas optimisation in updateRewards()	Gas Optimization	🟡 Low	Resolved
RPP-02	ReentrancyGuard not required	Gas Optimization	🟡 Low	Resolved
RPP-03	Addresses from gameAddresses should be cached	Gas Optimization	🟡 Low	Resolved
RPP-04	BZAI property should be immutable	Gas Optimization	🟡 Low	Resolved
RRF-04	Use an amount as a parameter in balancerToPvpReward() and balancerToWinPveReward()	Coding Style	🟡 Low	Resolved
RRF-05	Properties should be immutable	Gas Optimization	🟡 Low	Resolved

ID	Title	Category	Severity	Status
RRF-06	Use of Reentrancy guard	Gas Optimization	🟡 Low	Resolved
RWF-04	Use an amount as a parameter in balancerToPvpReward() and balancerToWinPveReward()	Coding Style	🟡 Low	Resolved
RWF-05	Properties should be immutable	Gas Optimization	🟡 Low	Resolved
RWR-02	Gas optimisation in updateRewards()	Gas Optimization	🟡 Low	Resolved
RWR-03	BZAI property should be immutable	Gas Optimization	🟡 Low	Resolved
RWT-02	getRewardsForTournament() transfers tokens to the owner account	Centralization / Privilege	🟡 Low	Acknowledge
RWT-03	Potential transfer of 0 tokens	Volatile Code	🟡 Low	Resolved
RWT-04	BZAI property should be immutable	Gas Optimization	🟡 Low	Resolved
SIG-02	Public functions could be external	Gas Optimization	🟡 Low	Acknowledge
SIG-03	Variables "owners" and "isOwner" stores similar data	Gas Optimization	🟡 Low	Acknowledge
TRC-05	setGameAddresses should be external	Gas Optimization	🟡 Low	Resolved
TRC-06	Addresses from gameAddresses should be cached	Gas Optimization	🟡 Low	Resolved
TRM-07	Event TrainingPurchase should emit more informations	Coding Style	🟡 Low	Resolved
TRM-08	Event CoachPaid should emit more informations	Coding Style	🟡 Low	Resolved
TRM-09	Unused property	Gas Optimization	🟡 Low	Resolved

ID	Title	Category	Severity	Status
TRM-10	maxDurationTraining should be constant	Gas Optimization	● Low	Resolved
TRM-11	setGameAddresses should be external	Gas Optimization	● Low	Resolved
ZFL-03	Gas optimisation in getUsedPowersByElement()	Gas Optimization	● Low	Resolved
ZFT-04	Multiplication on the result of a division	Language Specific	● Low	Resolved
ZFT-05	Event FightResult should emit more informations	Coding Style	● Low	Resolved
ZFT-06	Addresses from gameAddresses should be cached	Gas Optimization	● Low	Resolved
ZFT-07	Address of oracle contract should be cached	Gas Optimization	● Low	Resolved
ZFT-08	If; statement not required	Gas Optimization	● Low	Resolved
ZMT-04	Properties should be constant	Gas Optimization	● Low	Resolved
ZMT-05	Duplicate variables	Gas Optimization	● Low	Resolved
ZMT-06	Unnecessary loop	Gas Optimization	● Low	Resolved
ZNF-04	Shadows of existing variable name	Language Specific	● Low	Resolved
ZST-05	Unused import	Gas Optimization	● Low	Resolved
ZST-06	Addresses from gameAddresses should be cached	Gas Optimization	● Low	Resolved
ALC-05	Comment inconsistency	Volatile Code	● Information	Resolved
DWR-05	NonReentrant modifier on internal functions	Coding Style	● Information	Resolved
GLB-04	Coding practice	Coding Style	● Information	Resolved
GLB-05	Interface inheritance	Coding Style	● Information	Acknowledge

ID	Title	Category	Severity	Status
GLB-06	Do not cast address(0) to an interface	Coding Style	● Information	Resolved
LIQ-09	Transferred amount not validated	Logical Issue	● Information	Resolved
LOT-04	Max level cannot be reached in _getPotionLoot()	Logical Issue	● Information	Acknowledge
MKP-11	Typo in error message	Coding Style	● Information	Acknowledge
MKZ-07	Properties should be defined before constructor	Coding Style	● Information	Resolved
RCR-03	Invalid comment	Volatile Code	● Information	Resolved
TRM-12	coachDatas should be in CapWords style	Coding Style	● Information	Resolved
ZST-07	Events should be named using the CapWords style	Coding Style	● Information	Resolved

ALC-01 | Invalid power attribution

Category	Severity	Location	Status
Volatile Code	● High	AlchemyV1.sol: 158~160	Resolved

Description

In the function `useAlchemy()`, the condition to affect the power in the case of Mana mix is reversed and `_additionnalPoints` won't be affected correctly.

```
if(!_isManaMix){
    _powers[6] += _additionnalPoints * 100;
}
```

Recommendation

Reverse the condition.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

ALC-02 | Duplicate test in useAlchemy

Category	Severity	Location	Status
Volatile Code	● Medium	AlchemyV1.sol: 72; 103	Resolved

Description

The function `useAlchemy()` has a modifier `canUseZai()`, but also checks that `msg.sender` is the owner of the token.

Recommendation

Remove the ownership verification, if delegates are allowed to call this function, or remove the modifier if only the owner of the token is allowed.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change and removed the validation on `msg.sender`.

ALC-03 | Gas optimisation in useAlchemy

Category	Severity	Location	Status
Gas Optimization	🟡 Low	AlchemyV1.sol: 87~101	Resolved

Description

The following tests will be executed in unnecessary cases.

```
if (_usedPotions.length == 6) {
        require(z.manaMax == 10000, "Not enough manaMax");
  }
  if (_usedPotions.length == 5) {
        require(z.manaMax >= 8000, "Not enough manaMax");
  }
  if (_usedPotions.length == 4) {
        require(z.manaMax >= 6000, "Not enough manaMax");
  }
  if (_usedPotions.length == 3) {
        require(z.manaMax >= 4000, "Not enough manaMax");
  }
  if (_usedPotions.length == 2) {
        require(z.manaMax >= 2000, "Not enough manaMax");
  }
```

Recommendation

Use `if; else;` statements to simplify the function and optimise gas consumption.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

ALC-04 | Cache addresses from gameAddresses

Category	Severity	Location	Status
Gas Optimization	● Low	AlchemyV1.sol	Resolved

Description

The addresses returned by `gameAddresses` are not cached, thus requiring external calls in all the functions.

Recommendation

We suggest adding a method to cache / update the addresses from `gameAddresses` and store them as variables in the contract.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

ALC-05 | Comment inconsistency

Category	Severity	Location	Status
Volatile Code	● Information	AlchemyV1.sol	Resolved

Description

The chances to mint a chicken are set to 5% in the implementation, though the comment specifies that it should be 4%.

```
// 4% chance to mint a magical chicken
if (_getRandom(msg.sender, _manaUsed) % 100 <= 4) {
    IChicken(gameAddresses.getChickenAddress()).mintChicken(msg.sender)
}
```

Recommendation

Update the comment to match the code to improve maintainability.

Alleviation

`[UnblockLabs]` The client updated the code to limit the chance to 4%.

BZA-01 | No events emitted

Category	Severity	Location	Status
Language Specific	🟠 High	BandZaiAddresses.sol	Resolved

Description

No events are emitted when an address is changed, making it hard to monitor efficiently the contract off chain.

Recommendation

We recommend declaring and emitting corresponding events for all the essential state variables that are possible to be changed during runtime.

Alleviation

`[UnblockLabs]` The client implemented and raised an event in the new method `setAddress()`.

BZA-02 | Addresses upgradability

Category	Severity	Location	Status
Coding Style	● High	BandZaiAddresses.sol	Resolved

Description

Most addresses in the contract BandZaiAddresses cannot be updated.
This could seem like good protection for users, but in case of a contract bug, this can significantly reduce the token's value.

Recommendation

- Allow addresses to be updated.
- Reinforce protection with a time lock before activating the new addresses. (address update can only be effective after a certain period of time, like 24 or 48h.)
- Allow an address update to be cancelable.
- Add events to monitor the changes.
- Implement a pub/sub mechanism to enforce contracts that use `BandZaiAddresses` to be updated with the new addresses.

Alleviation

`[UnblockLabs]` The client implemented an update mecanism with a timelock of 1 days.
It is to be noted that the variable `contractName` should be an `enum` instead of a collection of `string`.

BZA-03 | Random number Oracle complexity

Category	Severity	Location	Status
Language Specific	● Medium	BandZaiAddresses.sol: 216~222	Resolved

Description

The contract randomly returns 2 different addresses for the random oracle. Those 2 contracts use a pretty similar implementation and this selection does not add any randomness to the final value.

```
function getOracleAddress() external view returns (address) {
    if (gasleft() % 2 == 0) {
        return oracleAddress1;
    } else {
        return oracleAddress2;
    }
}
```

Recommendation

We recommend using only 1 random oracle contract to increase maintainability.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

BZA-04 | Unused variable

Category	Severity	Location	Status
Gas Optimization	🟡 Low	BandZaiAddresses.sol: 21	Resolved

Description

The variable `teamAddress` is never set or used within the implementation of the contract.

Recommendation

Remove unused variables.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

BZT-01 | Ownership of BandZaiToken contract

Category	Severity	Location	Status
Centralization / Privilege	🟡 Low	BandZaiToken.sol	Resolved

Description

The contracts `BandZaiToken` inherits `Ownable` to allow the owner to withdraw `BZAI` tokens inadvertently transferred to the contracts.

Recommendation

This ownership privilege does not add any specific features to the protocol and the team should consider removing the `withdraw()` function and the ownership.
If this feature is required, we recommend taking an address of the `ERC20` token as a parameter to be able to retrieve other tokens than `BZAI`.

Alleviation

`[UnblockLabs]` The client opted to keep the ownership of the token and add the address of the token to recover as a parameter of the `withdraw()` function.

CHK-01 | Check Effects Interactions pattern violation

Category	Severity	Location	Status
Volatile Code	● High	ChickenNFT.sol: 49~61	Resolved

Description

In the function `mintChicken()`, the state is updated after the mint, allowing a potential reentrancy attack.

Recommendation

Move the call to `_safemint()` after the state update.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

CHK-02 | Duplicate functionality from base class

Category	Severity	Location	Status
Gas Optimization	● Medium	ChickenNFT.sol: 12~13	Resolved

Description

The counter `_tokenIds` duplicates the functionality already present in the base class `ERC721Enumerable` which already exposes `totalSupply()`. The value of `totalSupply` is always identical to `_tokenIds.current()`.

Recommendation

Remove the `_tokenIds` variable and use `totalSupply()` from base class.

Alleviation

`[UnblockLabs]` The client opted to use `ERC721` as the base class and manage the `totalSupply` locally.

CLB-01 | No events emitted

Category	Severity	Location	Status
Language Specific	🟠 Medium	ClaimBzai.sol	Resolved

Description

The following functions does not emit events to pass informations off chain:

- `setTgeBlock`
- `setAdvisorsVesting`
- `setPrivateVesting`
- `setPublicVesting`
- `setTgeBlock`
- `claimBZAIs`
- `withdrawUnassigned`
- `resetNFT`

Recommendation

We recommend declaring and emitting corresponding events for all the essential state variables that are possible to be changed during runtime.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

CLB-02 | resetNFT() does not clean data structures

Category	Severity	Location	Status
Gas Optimization	● Medium	ClaimBzai.sol: 83~88	Resolved

Description

The function `resetNFT()` burns the token but does not clean the data associated with it.

```
function resetNFT(uint256 tokenId) external onlyOwner{
    require(block.number <= tgeUnlockedBlock || tgeUnlockedBlock
== 0, "too late to change anything");
    _burn(tokenId);
    assigned -= _vestingNFT[tokenId].initialAmount;
    delete _vestingNFT[tokenId];
}
```

The following 3 variables associated with the token are not cleared:

- `mapping(uint256 => uint256[7]) _privateVestingAmount;`
- `mapping(uint256 => uint256[6]) _advisorsVestingAmount;`
- `mapping(uint256 => uint256[4]) _publicVestingAmount;`

Also, the function decrements the value of `_vestingNFT[tokenId].initialAmount` from `assigned` but the `initialAmount` is never set within the implementation of the contract.

Recommendation

Clear the data in the mapping to save gas.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

CLB-03 | Gas optimisation in setAdvisorsVesting()

Category	Severity	Location	Status
Gas Optimization	● Low	ClaimBzai.sol: 110~119	Resolved

Description

This algorithm will set the variable `_advisorsVestingAmount` 2 times when `i == 5`, and can be optimized to save gas.

```
110  for(uint256 i = 0 ; i < 6 ; ){
111      _advisorsVestingAmount[nftId][i] = claimablePart;
112
113      if(i == 5){
114          // add rest of division to last claim
115          _advisorsVestingAmount[nftId][i] += modulo;
116      }
117
118      unchecked { ++i ;}
119  }
```

Recommendation

Remove the loop
ie:

```
_advisorsVestingAmount[nftId][0] = claimablePart;
_advisorsVestingAmount[nftId][1] = claimablePart;
_advisorsVestingAmount[nftId][2] = claimablePart;
_advisorsVestingAmount[nftId][3] = claimablePart;
_advisorsVestingAmount[nftId][4] = claimablePart;
_advisorsVestingAmount[nftId][5] = claimablePart + modulo;
```

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

CLN-01 | Check Effects Interactions pattern Violation

Category	Severity	Location	Status
Volatile Code	● High	ClaimNFT.sol: 110~145; 115~119	Resolved

Description

In the function `claimAllNFTs()`, the state is updated after the NFT is minted, allowing a potential reentrancy attack.

```
115  for(uint256 i = 0 ; i < nurseries.length ;){
116      nursery.safeTransferFrom(address(this), msg.sender,
     nurseries[i]);
117      EnumerableSet.remove(myNurseries[msg.sender], nurseries[i]);
118      unchecked{ ++ i ; }
119  }
```

Recommendation

Update the state before transferring the tokens.
ie:

```
for(uint256 i = 0 ; i < nurseries.length ;){
  EnumerableSet.remove(myNurseries[msg.sender], nurseries[i]);
  nursery.safeTransferFrom(address(this), msg.sender, nurseries[i]);
  unchecked{ ++ i ; }
}
```

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

CLN-02 | Unbounded loop in claimAllNFTs

Category	Severity	Location	Status
Volatile Code	● High	ClaimNFT.sol: 110~145	Resolved

Description

The function `claimAllNFTs()` can overflow the block gas limit if many NFT are to be claimed, preventing the user from claiming his NFTs.

Recommendation

Add an upper limit to the number of NFT that can be claimed in a single call.

Alleviation

`[UnblockLabs]` The client opted to limit the claim to 2 NFTs per type, per call.

CLN-03 | No events emitted

Category	Severity	Location	Status
Language Specific	● Medium	ClaimNFT.sol	Resolved

Description

The following functions does not emit events to pass informations off chain:

- `setNFTs`
- `setNurseryOwner`
- `setTrainingOwner`
- `setLaboratoryOwner`
- `setTicketOwner`

Recommendation

We recommend declaring and emitting corresponding events for all the essential state variables that are possible to be changed during runtime.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.
It is to be noted that the event `OwnerSetted` should not use a string for `NftType` parameter but an `enum` or `integer` instead to minimize gas consumption.

```
event OwnerSetted(string indexed NftType, address indexed futurOwner,
uint256 indexed tokenId)
```

CLN-04 | Missing input validation

Category	Severity	Location	Status
Language Specific	🟡 Low	ClaimNFT.sol: 30~35	Resolved

Description

The function `setNFTs()` does not validate the inputs passed to the function.

Recommendation

Add a verification to validate that the addresses sent as parameters are not equal to `address(0)`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

CLT-01 | Previous assignation of tokens not take into account

Category	Severity	Location	Status
Logical Issue	● Medium	ClaimTeamAndMarketingBzai.sol	Resolved

Description

—

Recommendation

—

CLT-02 | No events emitted

Category	Severity	Location	Status
Language Specific	● Medium	ClaimTeamAndMarketingBzai.sol	Resolved

Description

The following functions does not emit events to pass informations off chain:

- `resetAlloc`
- `setTgeBlock`

Recommendation

We recommend declaring and emitting corresponding events for all the essential state variables that are possible to be changed during runtime.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

CLT-03 | Potential overflow in setMarketingVesting

Category	Severity	Location	Status
Volatile Code	● Low	ClaimTeamAndMarketingBzai.sol: 88	Resolved

Description

In the function `setMarketingVesting()`, the code used to test the amount can potentially overflow if `BZAI.balanceOf(address(this)) - assigned - _amount` is less than `0`.

```
require(BZAI.balanceOf(address(this)) - assigned - _amount > 0,
"To much assigned");
```

Recommendation

Update the test to never overflow.
ie:

```
require(BZAI.balanceOf(address(this)) >= assigned + _amount, "Too
much assigned");
```

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

CLT-04 | Check Effects Interactions pattern Violation

Category	Severity	Location	Status
Volatile Code	🟡 Low	ClaimTeamAndMarketingBzai.sol: 186~187	Resolved

Description

External calls should be the last ones operated.

Recommendation

Change:

```
186 | require(BZAI.transfer(msg.sender, _cleaned));
187 | require(assigned <= BZAI.balanceOf(address(this)));
```

to:

```
require(assigned <= BZAI.balanceOf(address(this)) - _cleaned);
require(BZAI.transfer(msg.sender, _cleaned));
```

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

CLT-05 | Redundant code

Category	Severity	Location	Status
Gas Optimization	🟡 Low	ClaimTeamAndMarketingBzai.sol	Resolved

Description

In the functions `claimMarketingBZAIs()` and `claimTeamBZAIs()`, the algorithm used to calculate `_claimable` and `_cleaned` is the same and will always return the same value, thus making the assertion `require(_claimable == _cleaned)` redundant and using unnecessary gas.

Recommendation

Remove the `require` statement.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

CLT-06 | Gas optimisation in setTeamVesting and setMarketingVesting

Category	Severity	Location	Status
Gas Optimization	🟡 Low	ClaimTeamAndMarketingBzai.sol: 64; 65; 69; 92	Resolved

Description

The functions `setTeamVesting()` and `setMarketingVesting()` manipulates the state variable `assigned` directly which cost more gas

Recommendation

Use a local scoped variable and assign the state variable `assigned` only once.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

DWR-01 | Missing input validation

Category	Severity	Location	Status
Volatile Code	● Medium	DailyWeeklyRanking.sol: 91~109	Resolved

Description

The function `setNickname()` does not validate the input passed as parameters.
A nickname with an empty or long size can be used without restriction.

Recommendation

Check the nickname's length.

Alleviation

`[UnblockLabs]` The client opted to limit the nickname length to 16 chars.

DWR-02 | Duplicate variable

Category	Severity	Location	Status
Gas Optimization	🟡 Low	DailyWeeklyRanking.sol: 17; 20	Resolved

Description

Both variables point to the same contract.

```
IPayments public IPay;
```

```
address public paymentAddress;
```

Recommendation

Remove `paymentAddress` and use `address(IPay)` when needed.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

DWR-03 | Gas optimisation in _updateDailyRanking()

Category	Severity	Location	Status
Gas Optimization	🟡 Low	DailyWeeklyRanking.sol: 375~421	Resolved

Description

The function `_updateDailyRanking()` can be optimized to save gas fees.

Recommendation

Refactor the algorythm to reduce gas used.

Alleviation

`[UnblockLabs]` The client opted to adaptthe function to reduce the gas consumption.

DWR-04 | Use bytes to compare strings

Category	Severity	Location	Status
Gas Optimization	● Low	DailyWeeklyRanking.sol: 100~103	Resolved

Description

The function `setNickname()` uses `abi.encodePacked` to compare strings.

```
if (
    keccak256(abi.encodePacked(addressToNickname[msg.sender]))
!=
    keccak256(abi.encodePacked(""))
)
```

Recommendation

It is recommended to use `keccak256(bytes(_string_))` to compare strings.
ie:

```
function compareStrings(string calldata a, string calldata b) public
returns (bool) {
  return keccak256(bytes(a)) == keccak256(bytes(b));
}
```

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

DWR-05 | NonReentrant modifier on internal functions

Category	Severity	Location	Status
Coding Style	● Information	DailyWeeklyRanking.sol: 283; 311	Resolved

Description

The functions `_payDailyWinners()` and `_payWeeklyWinners()` uses the modifier `nonReentrant`.
It is recommended to limit the use of this modifier to external functions.

> *Note that because there is a single `nonReentrant` guard, functions marked as `nonReentrant` may not call one another. This can be worked around by making those functions `private`, and then adding `external` `nonReentrant` entry points to them.*

Recommendation

Move the `nonReentrant` modifier to the `external` caller functions.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

DMZ-01 | Function canUseZai() does not check expired scholarships

Category	Severity	Location	Status
Volatile Code	● Medium	DelegateMyZai.sol: 250~270	Resolved

Description

The function `canUseZai()` still returns `true` for a delegate when a scholarship is expired. In the current implementation, this missing check has a limited impact but some functions like `updatePowers` can still be called by the delegate.

Recommendation

The function `canUseZai()` should check if scholarship is still active or not.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

DMZ-02 | ZaiAddress should be cached and immutable

Category	Severity	Location	Status
Coding Style	● Medium	DelegateMyZai.sol	Resolved

Description

The contract `DelegateMyZai` is linked to a specific NFT collection. If `ZaiAddress` is changed, the data stored by `_delegateDatas` will be corrupted.

Recommendation

`ZaiAddress` should be cached upon creation and not be updatable.

Alleviation

`[UnblockLabs]` The client opted to remove the reference to the `ZaiAddress`.

DMZ-03 | Pagination should be done by the caller

Category	Severity	Location	Status
Coding Style	🟡 Low	DelegateMyZai.sol	Resolved

Description

The following functions uses a fixed number of items per page returned by `getNumberOfDelegationPages()` to return paginated data:

- `getZaisInDelegation()`
- `getDelegatedToScholar()`

Recommendation

Allow the caller to specify dynamically the number of items returned and the startIndex. (ie by using a `skip` and `take` parameters)

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

EGG-01 | Check Effects Interactions pattern Violation

Category	Severity	Location	Status
Volatile Code	● High	EggsNFT.sol: 57~59	Resolved

Description

In the function `mintEgg()`, the call to `_safeMint` is done prior to updating the state, allowing a potential reentrancy attack.

```
57  _safeMint(_to, _newItemId);
58  _stateIndex[_newItemId] = _state;
59  _maturityTimestamp[_newItemId] = block.timestamp +
    _maturityDuration;
```

Recommendation

Update the state before calling `_safeMint()`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

EGG-02 | No events emitted

Category	Severity	Location	Status
Language Specific	● Medium	EggsNFT.sol: 46; 75	Resolved

Description

The following functions does not emit events to pass informations off chain:

- `setHourAccelerationPrice`
- `updateMaturity`

Recommendation

We recommend declaring and emitting corresponding events for all the essential state variables that are possible to be changed during runtime.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

EGG-03 | updateMaturity is never called

Category	Severity	Location	Status
Volatile Code	● Low	EggsNFT.sol: 75~77	Resolved

Description

The function `updateMaturity()` is never called by any contracts of the protocol, though it as an `onlyAuth` modifier.

Recommendation

Remove the function if not useful.

Alleviation

`[UnblockLabs]` The client opted to remove the function.

EGG-04 | Missing owner validation in coverEggWithChicken

Category	Severity	Location	Status
Volatile Code	🟡 Low	EggsNFT.sol: 150	Resolved

Description

The function `coverEggWithChicken()` allows covering an `EggNFT` from another owner. Any `ChickenNFT` can be used to cover any `EggNFT`, even if the egg's owner does not want to cover his token.

Recommendation

Check if the egg and chicken owners are the same.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

EGG-05 | Missing input validation

Category	Severity	Location	Status
Volatile Code	● Low	EggsNFT.sol: 112~115	Resolved

Description

The function `burnBzaiForClaimZai()` does not validate that `_toBurn > 0`.

Recommendation

Verify if `_toBurn > 0` and skip transfer and burn when `false`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

EGG-06 | Shadows of existing variable name

Category	Severity	Location	Status
Language Specific	🟡 Low	EggsNFT.sol: 81; 110	Resolved

Description

The functions `claimMatureZai()` and `burnBzaiForClaimZai()` use `_name` as a parameter which overrides `_name` variable from `ERC721Enumerable.`

Recommendation

Change the `_name` parameter to `name_`.

Alleviation

`[UnblockLabs]` The client opted to change the name of the variable to `_zaiName`.

EGG-07 | No added value in _isCover

Category	Severity	Location	Status
Gas Optimization	🟡 Low	EggsNFT.sol: 29~30	Resolved

Description

The state variables `_isCover` and `_isCoverBy` stores duplicate information.

Recommendation

Remove `_isCover` and use `_isCoverBy != address(0)`

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

EGG-08 | Gas optimisation in coverEggWithChicken

Category	Severity	Location	Status
Gas Optimization	● Low	EggsNFT.sol: 161	Resolved

Description

In the function `coverEggWithChicken()`, the require condition is evaluated after updating the state.

```
require(_originMaturity > _maturityTimestamp[_eggId], "Doesn't
need to cover this egg");
```

Recommendation

Check requirements at the beginning of the method, when possible, to save gas.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

GLD-01 | delegateNFTs should use safeTransferFrom and implement onErc721Received

Category	Severity	Location	Status
Volatile Code	● Medium	GuildeDelegation.sol: 52	Resolved

Description

The contract `GuildeDelegation` does not implement `onErc721Received` whereas it manipulates NFTs.

Recommendation

Inherit `ERC721Holder` in `GuildeDelegation` and use the function `safeTransferFrom()` instead of `transferFrom()` while transferring NFTs.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

GLD-02 | Possible underflow exception in delegateNFTs()

Category	Severity	Location	Status
Volatile Code	● Medium	GuildeDelegation.sol: 48; 63	Resolved

Description

The function `delegateNFTs()` can potentially raise an underflow exception if `_percentageForScholars + _platformFees > 100`.

```
48  require(_percentageForScholars > 0 && _percentageForScholars <
    100, "Bad percentage");
```

```
63  g.percentageForGuilde = 100 - _percentageForScholars -
    _platformFees;
```

Recommendation

Add a verification to limit the the range of `_percentageForScholars` and prevent underflows.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

GLD-03 | Unbounded loop in _getScholarNFTs and _getGuildNFTs

Category	Severity	Location	Status
Gas Optimization	🟠 Medium	GuildeDelegation.sol: 96~98; 113~115	Resolved

Description

The functions `_getScholarNFTs()` and `_getGuildNFTs()` loop through many potential NFTs. It can lower the user experience or reach a gas limit.

Recommendation

Allow the caller to specify dynamically the number of items returned and the startIndex. (ie by using a `skip` and `take` parameters)

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

GLD-04 | Unused ERC721Holder import

Category	Severity	Location	Status
Volatile Code	● Low	GuildeDelegation.sol: 5	Resolved

Description

`ERC721Holder` is imported, but not used.

Recommendation

Inherit `ERC721Holder` in `GuildeDelegation` or remove useless import.

Alleviation

`[UnblockLabs]` The client opted to inherit `ERC721Holder`.

GLD-05 | No added value in _getRentingDatas

Category	Severity	Location	Status
Gas Optimization	🟡 Low	GuildeDelegation.sol: 34~36; 38~40	Resolved

Description

The internal function `_getRentingDatas()` is called only once within the implementation of the contract.

Recommendation

Inline the method in the caller function `getRentingDatas()`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

INT-01 | Structure optimisation

Category	Severity	Location	Status
Gas Optimization	🟡 Low	Interfaces.sol: 9~15	Resolved

Description

Some structures could be optimised by changing the type of the variables used.

```
struct Powers {
    uint256 water;
    uint256 fire;
    uint256 metal;
    uint256 air;
    uint256 stone;
}
```

Recommendation

Use appropriate variable types based on the max values expected (`uint8`, `uint32`, ...) to optimise gas consumption.

Alleviation

`[UnblockLabs]` The client modified the structures to optimize the gas consumption.

IPF-01 | EnumerableSet does not guarantee order

Category	Severity	Location	Status
Volatile Code	🟠 High	IpfsIdStorage.sol	Resolved

Description

The function `_getGodId()` uses `EnumerableSet` to sort `_freeIds` and check if a god was minted.
As stated in OpenZeppelin's documentation of `EnumerableSet`, the order is not guarantee.

> *Note that there are no guarantees on the ordering of values inside the array, and it may change when more values are added or removed.*

Recommendation

Change the algorithm to not rely on the order of the collection.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

IPF-02 | No added value in _getIdsLength

Category	Severity	Location	Status
Gas Optimization	🟡 Low	IpfsIdStorage.sol: 178~184	Resolved

Description

The internal function `_getIdsLength()` is called only once within the implementation of the contract.

Recommendation

Inline the method in the caller function `getIdsLength()`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

LAM-01 | EnumerableSet does not guarantee order

Category	Severity	Location	Status
Volatile Code	🟠 High	LaboManagement.sol	Resolved

Description

The function `getLast10soldPotions()` relies on `EnumerableSet` to order the list.
As stated in OpenZeppelin's documentation of `EnumerableSet`, the order is not guarantee.

> *Note that there are no guarantees on the ordering of values inside the array, and it may change when more values are added or removed.*

Recommendation

Change the algorithm to not rely on the order of the collection.

Alleviation

`[UnblockLabs]` The client opted to remove the function.

LAM-02 | Random revert in workInASpot()

Category	Severity	Location	Status
Volatile Code	● Medium	LaboManagement.sol: 128; 130~139	Resolved

Description

The function `workInASpot()` uses a weak anti-bot solution and revert randomly in certain conditions.

```
// preventing bot attack by randomize a revert transaction
during first hour after 24h of work for a Zai
```

Genuine requests from users could also be rejected.

Recommendation

Use more robust criterias to detect antibots like the number of transactions per time slot, or protect with a blacklist mechanism.

Alleviation

`[UnblockLabs]` The client opted to verify that `tx.origin == msg.sender` to prevent execution from smart contracts.

LAM-03 | Missing input validation

Category	Severity	Location	Status
Volatile Code	● Medium	LaboManagement.sol: 265~271	Resolved

Description

The function `createAndSellPotion` does not check that `_quantity > 0` and `_power > 0` making the call potentially useless.

Recommendation

Validate the parameters passed as input to the function.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

LAM-04 | No events emitted

Category	Severity	Location	Status
Language Specific	● Medium	LaboManagement.sol	Resolved

Description

The following functions does not emit events to pass informations off chain:

- `setPointCreditCost`
- `setMaxCredit`
- `setWorkingSpotPrice`
- `setSpotPrice`
- `setminimumTrainingPrice`

Recommendation

We recommend declaring and emitting corresponding events for all the essential state variables that are possible to be changed during runtime.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.
It is to be noted that the event `MetricsChanged` should not use a `string` for the parameter `metricType` but an `enum` or an `integer` instead to minimise gas consumption.

LAM-05 | PotionSold event should expose potionId and buyer address

Category	Severity	Location	Status
Coding Style	● Medium	LaboManagement.sol: 20	Resolved

Description

The event `PotionSold` only exposes `labOwner` and `price`. This can make the event hard to use off-chain to recreate the state of the contract.

Recommendation

`PotionSold` event should expose `potionId` and *buyer address* as indexed params.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

LAM-06 | Unbounded loop in getUnsoldPotions()

Category	Severity	Location	Status
Gas Optimization	🟠 Medium	LaboManagement.sol: 330~337	Resolved

Description

The function `getUnsoldPotions()` enumerates through all the unsold potions without limiting the size of the enumeration and can result in an out of gas exception.

Recommendation

Allow the caller to specify dynamically the number of items returned and the startIndex. (ie by using a `skip` and `take` parameters)

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

LAM-07 | Potions from burnt Laboratory are still listed for sale

Category	Severity	Location	Status
Volatile Code	🟡 Low	LaboManagement.sol	Resolved

Description

Potions from a burnt laboratory are still listed for sale.

Recommendation

Burn all the potions from a laboratory when it's burnt.

Alleviation

`[UnblockLabs]` The client opted to add a verification that the potions can't be bought of the laboratory was burned.

LAM-08 | PotionSold event should be emitted after the transfer

Category	Severity	Location	Status
Coding Style	● Low	LaboManagement.sol: 443~457	Resolved

Description

`PotionSold` event is emitted before the effective NFT transfer.

```
emit PotionSold(p.seller, p.listingPrice);
...
IERC721(_potionAddress).transferFrom(
    address(this),
    msg.sender,
    _potionId
);
```

Recommendation

Emit the event after the transfer.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

LAM-09 | Addresses from gameAddresses should be cached

Category	Severity	Location	Status
Gas Optimization	🟡 Low	LaboManagement.sol: 55; 64; 99; 111; 115; 131	Resolved

Description

Adresses from `gameAddresses` are used in different functions within the implementation of the contract. Each call to `gameAddresses` requires an external call that could be avoided by caching the addresses used.

Recommendation

Cache the addresses returned by `gameAddresses`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

LAM-10 | Unused property

Category	Severity	Location	Status
Gas Optimization	🟡 Low	LaboManagement.sol: 30	Resolved

Description

The property `BZAI` is declared but never used within the implementation of the contract.

Recommendation

Remove unused properties.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

LAM-11 | setWorkingSpotPrice should be external

Category	Severity	Location	Status
Gas Optimization	🟡 Low	LaboManagement.sol: 87~89	Resolved

Description

`setWorkingSpotPrice` is public but not called within the contract implementation.

Recommendation

Set the function's visibility to `external`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

LAM-12 | Structure/Mapping optimisations

Category	Severity	Location	Status
Gas Optimization	🟡 Low	LaboManagement.sol	Resolved

Description

The following properties all uses a mapping with an identical key set to the lab NFT token id.

- `laboratoryRevenues`
- `potionsCredits`
- `laboratoryRevenues`
- `zaiNumberOfWork`
- `workingSpot`
- `employees`

This uses more gas than an optimised structure.

Recommendation

Create a `struct` to hold all the informations in 1 mapping.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

LAM-13 | createdPotionsForLab can be recreated offchain from events

Category	Severity	Location	Status
Gas Optimization	🟡 Low	LaboManagement.sol: 39	Resolved

Description

The property `createdPotionsForLab` is stored on-chain whereas the state is not used internally.

Recommendation

Recreate the equivalent data off-chain based on events emitted by the contract.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

LAB-01 | Missing onlyAuth modifier in updateCreditLastUpdate

Category	Severity	Location	Status
Volatile Code	● High	Laboratory.sol: 97~100	Resolved

Description

The function `updateCreditLastUpdate` can be called by anyone. This would prevent other players from claiming their tokens.

Recommendation

Add the `onlyAuth` modifier to `updateCreditLastUpdate`

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

LAB-02 | No restriction on _preMintNumber in the constructor

Category	Severity	Location	Status
Volatile Code	● Medium	Laboratory.sol: 22~25	Resolved

Description

`_preMintNumber` is not limited and the deployment can overflow the block gas limit if the value is too high.

Recommendation

Create an independent function out of the constructor to pre-mint items and pass the quantity as a parameter of the function.

Alleviation

`[UnblockLabs]` The client opted to limit the pre minted NFTs to 29.

LAB-03 | Addresses from gameAddresses should be cached

Category	Severity	Location	Status
Gas Optimization	● Medium	Laboratory.sol: 36; 37	Resolved

Description

Adresses from `gameAddresses` are used in different functions within the implementation of the contract. Each call to `gameAddresses` requires an external call that could be avoided by caching the addresses used.

Recommendation

Cache the addresses returned by `gameAddresses`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

LAB-04 | Duplicate functionality

Category	Severity	Location	Status
Gas Optimization	🟡 Low	Laboratory.sol: 10~11	Resolved

Description

The counter `_tokenIds` duplicates the functionality implemented by the parent class `ERC721Enumerable`.
The value of `totalSupply()` will always be identical to `_tokenIds.current()`.

Recommendation

Remove `_tokenIds` and `use totalSupply()` from `ERC721Enumerable`.

Alleviation

`[UnblockLabs]` The client opted to change the implementation to inherit directly `ERC721`.

LVL-01 | Missing validation in getRandomZaiFromLevel

Category	Severity	Location	Status
Volatile Code	● Medium	LevelStorage.sol: 55~83	Acknowledge

Description

The function `getRandomZaiFromLevel` does not check if a fighter is at the requested level. If no fighter is available for that level, this will crash the index.

Recommendation

Check that the requested level has more than one fighter.
ie:

```
require(levelFighters[_level].length() > 1, "Invalid level")
```

Alleviation

`[UnblockLabs]` The client opted to keep the implementation as is.
`[BandZai]` As challengers are automatically created when a Zai enters a level, this case will never happen. Adding this check will increase gas consumption.

LVL-02 | No events emitted

Category	Severity	Location	Status
Coding Style	● Medium	LevelStorage.sol	Resolved

Description

The following functions does not emit events to pass informations off chain:

- `addFighter`
- `removeFighter`
- `setGameAddresses`
- `updateAddresses`

Recommendation

We recommend declaring and emitting corresponding events for all the essential state variables that are possible to be changed during runtime.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

LIQ-01 | Invalid variable decrement

Category	Severity	Location	Status
Logical Issue	● High	LiquidityMining.sol: 127; 148	Resolved

Description

The value of `remainingBZAIReward` is decremented in the `deposit()` and `withdraw()` functions. This will lead to blocking tokens on the contract that could not be distributed.

```
remainingBZAIReward -= amount;
```

Recommendation

`remainingBZAIReward` should not be decremented in `deposit()` and `withdraw()`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

LIQ-02 | getMiningStarted returns true when not started

Category	Severity	Location	Status
Volatile Code	● Medium	LiquidityMining.sol: 59~61	Resolved

Description

The name of the method suggests the opposite of what the code does. This could lead to the wrong usage of the method.

```
function getMiningStarted() external view returns (bool) {
    return liquidityMining.lastRewardBlock == 0;
}
```

Recommendation

Change the method name to reflect what the code does.

Alleviation

`[UnblockLabs]` The client opted to rename the function to `isMiningStarted()`.

LIQ-03 | No events emitted

Category	Severity	Location	Status
Coding Style	● Medium	LiquidityMining.sol: 54~57	Resolved

Description

The following function do not emit events to pass informations off chain:

- `startMining`

Recommendation

We recommend declaring and emitting corresponding events for all the essential state variables that are possible to be changed during runtime.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

LIQ-04 | Missing input validation

Category	Severity	Location	Status
Volatile Code	🟡 Low	LiquidityMining.sol: 37	Resolved

Description

The function `setTokensAddress()` does not validate that parameters passed to the function are not equal to address(0).

Recommendation

Check that both `_bzai` and `_lpToken` parameters are not equal to address(0).

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

LIQ-05 | ReentrancyGuard not used

Category	Severity	Location	Status
Gas Optimization	🟡 Low	LiquidityMining.sol: 10	Resolved

Description

The contract `LiquidityMining` inherits from `ReentrancyGuard` but it's not used.

Recommendation

Remove `ReentrancyGuard` inheritance and import.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

LIQ-06 | Gas optimisation in updatePool()

Category	Severity	Location	Status
Gas Optimization	🟡 Low	LiquidityMining.sol: 64~68; 88~95; 93	Resolved

Description

The condition `block.number <= liquidityMining.lastRewardBlock` can never be reached thanks to the requirement above:

```
88  require(
89      liquidityMining.lastRewardBlock > 0 &&
90          block.number >= liquidityMining.lastRewardBlock,
91      "Mining not yet started"
92  );
93  if (block.number <= liquidityMining.lastRewardBlock) {
94      return;
95  }
```

Recommendation

Remove the `if` condition in line 93.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

LIQ-07 | Gas optimisation in claim()

Category	Severity	Location	Status
Gas Optimization	🟡 Low	LiquidityMining.sol: 117; 171	Resolved

Description

The state variable `pendingRewards` is set twice in the function claim(), which costs an extra gas that can be avoided.

Recommendation

Set `pendingRewards` only once and use local variables for local computations.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

LIQ-08 | BZAIPerBlock should be constant

Category	Severity	Location	Status
Gas Optimization	🟡 Low	LiquidityMining.sol: 27	Resolved

Description

`BZAIPerBlock` is never updated within the implementation of the contract and should be declared constant.

Recommendation

Declare `BZAIPerBlock` as a `constant`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

LIQ-09 | Transferred amount not validated

Category	Severity	Location	Status
Logical Issue	● Information	LiquidityMining.sol: 121~126	Resolved

Description

In the function `deposit()`, the quantity of tokens transferred is not verified by the contract.
This can lead to an overestimate of amount received if, for example, `lpToken` has a transfer tax.

```
liquidityMining.lpToken.safeTransferFrom(
    address(msg.sender),
    address(this),
    amount
);
user.amount += amount;
```

Recommendation

Check the balance before and after transferring the tokens to calculate the amount transferred.

```
uint256 balanceBefore =
liquidityMining.lpToken.balanceOf(address(this));
liquidityMining.lpToken.safeTransferFrom(
    address(msg.sender),
    address(this),
    amount
);
uint256 balanceAfter =
liquidityMining.lpToken.balanceOf(address(this));
amount = balanceAfter - balanceBefore;
user.amount += amount;
...
```

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

LOT-01 | Addresses from gameAddresses should be cached

Category	Severity	Location	Status
Gas Optimization	● Medium	LootProgress.sol: 35; 85; 165; 182; 198	Resolved

Description

Adresses from `gameAddresses` are used in different functions within the implementation of the contract. Each call to `gameAddresses` requires an external call that could be avoided by caching the addresses used.

Recommendation

Cache the addresses returned by `gameAddresses`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

LOT-02 | Use of string parameter in event NewLootResult

Category	Severity	Location	Status
Gas Optimization	🟡 Low	LootProgress.sol: 31	Resolved

Description

`NewLootResult` uses `string` type for `lootType` whereas the values are known in advance.

Recommendation

Encode `lootType` with integer type.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

LOT-03 | Loop can be replaced by a division

Category	Severity	Location	Status
Gas Optimization	🟡 Low	LootProgress.sol: 147~151	Resolved

Description

The while loop can be replaced by a division to simplify the code and optimise gas consumption.

```
uint256 _tens = 0;
while(_weekNumber > 10){
    _weekNumber -= 10;
    _tens += 1;
}
```

Recommendation

Replaced the while loop by a division.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

LOT-04 | Max level cannot be reached in _getPotionLoot()

Category	Severity	Location	Status
Logical Issue	● Information	LootProgress.sol: 171	Acknowledge

Description

The `_power` affected in the function `_getPotionLoot()` is assigned between `_minLevel` and `_maxLevel - 1`. Max level cannot be reached.

```
uint256 _power = _minLevel + (r[i] % (_maxLevel - _minLevel));
```

Recommendation

Update the computation to reach the max level.

Alleviation

`[UnblockLabs]` The client opted to keep the implementation as is.

MKP-01 | bidForNft() does not check for revenues

Category	Severity	Location	Status
Logical Issue	● High	MarketPlace.sol: 170~179	Resolved

Description

The function `bidForNft` checks for token allowance whereas the bid could be paid with rewards and/or revenues.
Revenues are ignored to check the balance.

Recommendation

Remove the allowance requirement and add revenues to the balance check.

Alleviation

`[UnblockLabs]` The client opted to include the available revenue in the balance.

MKP-02 | Approval can be restricted to the token only in sellNft()

Category	Severity	Location	Status
Centralization / Privilege	● Medium	MarketPlace.sol: 139	Resolved

Description

The function `sellNft()` checks approval for the entire collection, whereas it just needs to be approved for the token.

Recommendation

Allow the user to approve only the selected NFT token if desired.
ie:

```
require(
    I.isApprovedForAll(msg.sender, address(this)) ||
I.getApproved(_nftId) == address(this),
    "Need to approve the NFT"
);
```

Alleviation

`[UnblockLabs]` The client opted to implement the verification on a per token basis.

MKP-03 | NFT can be listed multiple times in sellNft()

Category	Severity	Location	Status
Volatile Code	● Medium	MarketPlace.sol: 131~157	Resolved

Description

The function `sellNft()` does not validate if an NFT is already listed and can result in an NFT listed multiple times.

Recommendation

Add a check to require that the NFT is not already listed for sale.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

MKP-04 | User can bid on his own NFT in bidForNft()

Category	Severity	Location	Status
Volatile Code	● Low	MarketPlace.sol: 164~199	Resolved

Description

An owner can bid on it's own NFT.

Recommendation

Add a verification to require that the bidder is not the owner of the token.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

MKP-05 | Check Effects Interactions pattern Violation in buyNft()

Category	Severity	Location	Status
Volatile Code	🟡 Low	MarketPlace.sol: 278~283	Resolved

Description

In the function `buyNft()` transfer of the NFT is done before updating the state.

```
IERC721(offer.nftAddress).transferFrom(
    offer.nftOwner,
    msg.sender,
    offer.nftId
);
delete _offers[_offerId];
```

Recommendation

Follow the check effects interactions pattern and update the state before transferring the token.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

MKP-06 | Check Effects Interactions pattern Violation in acceptBid()

Category	Severity	Location	Status
Volatile Code	● Low	MarketPlace.sol: 226~238	Resolved

Description

State change is done before checking the params in the function `acceptBid()`.

```
delete _offers[_offerId];

address payments = gameAddresses.getPaymentsAddress();
IPayments IPay = IPayments(payments);

require(
    BZAI.allowance(offer.offeredBy, payments) >= offer.bidValue,
    "Contract has been unapproved by bider"
);
require(
    block.number <= offer.creatingBlock + offerDuration,
    "Offers only avalaible 3 days"
);
```

Recommendation

Follow the check effects interactions pattern and verify the requirements before updating the state.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

MKP-07 | offerDuration is initialised from non-constant variable

Category	Severity	Location	Status
Language Specific	🟡 Low	MarketPlace.sol: 24~25	Resolved

Description

The variable offerDuration is initialised inline with non constant variables:

```
uint256 public blockPerDay = 43200;
uint256 public offerDuration = 3 * blockPerDay;
```

Recommendation

Move the initialisation of the variable inside the constructor of the contract.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change and not rely on a fixed value of blocks per day.

MKP-08 | Unnecessarily import of EnumerableSet.sol

Category	Severity	Location	Status
Gas Optimization	● Low	MarketPlace.sol: 4	Resolved

Description

`EnumerableSet.sol` is imported, whereas it is never used within the implementation of the contract.

Recommendation

Remove the import.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

MKP-09 | BZAI property should be immutable

Category	Severity	Location	Status
Gas Optimization	🟡 Low	MarketPlace.sol: 12	Resolved

Description

The property `BZAI` is never changed within the implementation of the contract and should be declared as `immutable`.

Recommendation

Declare the property `BZAI` as `immutable`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

MKP-10 | _myProposals can be recreated off-chain

Category	Severity	Location	Status
Gas Optimization	🟡 Low	MarketPlace.sol: 35	Resolved

Description

The variable _myProposals is never used within the implementation of the contract. This state can be recreated off-chain to optimise the gas consumption.

Recommendation

Use events to compute and monitor the list of proposals off-chain.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change and removed the variable.

MKP-11 | Typo in error message

Category	Severity	Location	Status
Coding Style	● Information	MarketPlace.sol: 242	Acknowledge

Description

There is a typo in the following error message:

```
"bider hasn't enough founds"
```

Recommendation

Fix the typo, ie `"bidder hasn't enough funds"`.

Alleviation

`[UnblockLabs]` The client decided to keep the error message as is.

MKZ-01 | _calculateDutchPrice reduction does not work within a day

Category	Severity	Location	Status
Logical Issue	● High	MarketZai.sol: 346~349	Resolved

Description

`_calculateDutchPrice` reduction does not work within a day. When inlining calculation, the reduction within a day is not applied.

```
346 //reduce nber of block past in the day multiply by value to
    reduce per block
347 _priceToReturn =
348     _priceToReturn -
349     (_pastBlockInCurrentDay * _toReducePerBlock);
```

Recommendation

Check the algorithm.

Alleviation

`[UnblockLabs]` The client adapted the algorithm to handle the reduction within a day.

MKZ-02 | No events emitted

Category	Severity	Location	Status
Coding Style	● Medium	MarketZai.sol: 61; 66; 203	Resolved

Description

The following function do not emit events to pass informations off chain:

- `setBlockPerDay`
- `setGameAddresses`
- `setMultiplicators`

Recommendation

We recommend declaring and emitting corresponding events for all the essential state variables that are possible to be changed during runtime.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

MKZ-03 | Addresses from gameAddresses should be cached

Category	Severity	Location	Status
Gas Optimization	🟠 Medium	MarketZai.sol: 73; 74; 89; 91; 261; 282; 359; 368	Resolved

Description

Adresses from `gameAddresses` are used in different functions within the implementation of the contract. Each call to `gameAddresses` requires an external call that could be avoided by caching the addresses used.

Recommendation

Cache the addresses returned by `gameAddresses`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

MKZ-04 | duplicate function "setBlockPerday" and "setBlockPerDay"

Category	Severity	Location	Status
Gas Optimization	🟡 Low	MarketZai.sol: 66~68; 217~219	Resolved

Description

The functions `setBlockPerday()` and `setBlockPerDay()` both have an identical implementation and a similar name.

Recommendation

Remove one of the two functions.

Alleviation

`[UnblockLabs]` The client opted to remove both functions and not rely on a fixed block count per day.

MKZ-05 | BZAI property should be immutable

Category	Severity	Location	Status
Gas Optimization	🟡 Low	MarketZai.sol: 20	Resolved

Description

The property `BZAI` is never changed within the implementation of the contract and should be declared as `immutable`.

Recommendation

Declare the property `BZAI` as `immutable`.

Alleviation

`[UnblockLabs]` The client opted to remove the property as it was unused.

MKZ-06 | Gas optimisation in _randomMint() and _getZaiPrice()

Category	Severity	Location	Status
Gas Optimization	🟡 Low	MarketZai.sol: 182~197; 289~306	Resolved

Description

The functions `_randomMint()` and `_getZaiPrice()` use only simple `if` statements.

Recommendation

Use `if` / `else` statements to optimise gas consumption.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

MKZ-07 | Properties should be defined before constructor

Category	Severity	Location	Status
Coding Style	● Information	MarketZai.sol: 41; 49~59	Resolved

Description

Some properties are defined after the constructor.

Recommendation

Properties should be defined before the constructor.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

SIG-01 | Upgradability of signers and required confirmations

Category	Severity	Location	Status
Volatile Code	🟠 High	MultiSigWallet.sol	Acknowledge

Description

`MultiSigWallet` does not allow the addition or removal of an owner. Also the number of required confirmations can't be changed. Both limitations can lead to security issues In case of a lost or compromised private key from one of the owners.

Recommendation

Allow managing the owners and required number of confirmations. Consider using a well know implementation like Gnosis Safe.

Alleviation

`[UnblockLabs]` The client acknowledged the issue.
`[BandZai]` We will probably use Gnosis Safe.

SIG-02 | Public functions could be external

Category	Severity	Location	Status
Gas Optimization	🟡 Low	MultiSigWallet.sol	Acknowledge

Description

The following functions are declared with a public visibility but are never called within the implementation of the contract:

- `confirmTransaction`
- `executeTransaction`
- `revokeConfirmation`

Recommendation

Consider setting the visibility of those functions to external to optimise gas consumption.

Alleviation

`[UnblockLabs]` The client acknowledged the issue.
`[BandZai]` We will probably use Gnosis Safe.

SIG-03 | Variables "owners" and "isOwner" stores similar data

Category	Severity	Location	Status
Gas Optimization	🟡 Low	MultiSigWallet.sol: 17~18	Acknowledge

Description

Both variable `owners` and `isOwner` stores similar data.

Recommendation

Remove `isOwner` and use the method `owners.contains(address)` when required.

Alleviation

`[UnblockLabs]` The client acknowledged the issue.
`[BandZai]` We will probably use Gnosis Safe.

NRS-01 | Check Effects Interactions pattern Violation

Category	Severity	Location	Status
Volatile Code	● High	Nursery.sol: 78~83	Resolved

Description

In the function `mintNursery()` transfer of the NFT is done before updating the state.

```
78  _safeMint(_to, _newItemId);
79  ZaiStruct.EggsPrices storage e = eggsPrices[_newItemId];
80  e.bronzePrice = _prices[0];
81  e.silverPrice = _prices[1];
82  e.goldPrice = _prices[2];
83  e.platinumPrice = _prices[3];
```

Recommendation

Follow the check effects interactions pattern and update the state before transferring the token.

Alleviation

`[UnblockLabs]` The client opted to remove the function.

NRS-02 | Pre-minting should not be done in the constructor of the contract

Category	Severity	Location	Status
Volatile Code	🟠 Medium	Nursery.sol: 37~50	Acknowledge

Description

The contract `Nursery.sol` could fail being deployed if the number of pre-minted NFT is too high.

Recommendation

Create an independent function out of the constructor to pre-mint items.

Alleviation

`[UnblockLabs]` The client kept the implementation as is.
`[BandZai]` PreMint Number will be 15 and won't failed.

NRS-03 | Unused variables

Category	Severity	Location	Status
Coding Style	● Low	Nursery.sol: 19~24	Resolved

Description

The variable `_maxPrices` is not used within the contract. It is used only by `NurseryManagement.sol` to when verifying the price.

Recommendation

Do not mix the responsibilities of the contracts. Move the variable `_maxPrices` to `NurseryManagement.sol` or use it in the function `_pricesOk()`.

Alleviation

`[UnblockLabs]` The client opted to move the variable in `NurseryManagement.sol`.

NRS-04 | Unused property

Category	Severity	Location	Status
Gas Optimization	🟡 Low	Nursery.sol: 11	Resolved

Description

The property `BZAI` is declared but never used within the implementation of the contract.

Recommendation

Remove unused properties.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

NRS-05 | Unused function

Category	Severity	Location	Status
Gas Optimization	🟡 Low	Nursery.sol: 69~86	Resolved

Description

The function `mintNursery()` is never called by any contracts whereas it as an `onlyAuth` modifier.

Recommendation

Remove the function if not used.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

NRM-01 | Invalid maturity date

Category	Severity	Location	Status
Logical Issue	● High	EggsNFT.sol: 59 NurseryManagement.sol: 235~239	Resolved

Description

The maturity date passed to `mintEgg` in the function `reserveNextEgg()` is invalid as it add `block.timestamp`, though it already done in `mintEgg().`

```
IEggs(gameAddresses.getEggsAddress()).mintEgg(
    _to,
    state,
    block.timestamp + maturities[state] * 2
);
```

In Eggs.NFT.sol:

```
_maturityTimestamp[_newItemId] = block.timestamp +
_maturityDuration;
```

Recommendation

Remove `block.timestamp`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

NRM-02 | No events emitted

Category	Severity	Location	Status
Language Specific	● Medium	NurseryManagement.sol	Resolved

Description

The following functions does not emit events to pass informations off chain:

- `setMaturityDurations`
- `setMinPrices`
- `setMaxPrices`

Recommendation

We recommend declaring and emitting corresponding events for all the essential state variables that are possible to be changed during runtime.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

NRM-03 | Mix of responsibility with base class

Category	Severity	Location	Status
Coding Style	● Medium	NurseryManagement.sol: 12; 83~93	Resolved

Description

`_minPrices` and `_maxPrices` should not be accessed in NurseryManagement. `NurseryManagement` should not inherit from `Nursery.`

Recommendation

Remove inheritance to `Nursery` and access to public properties only.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

NRM-04 | Unused property

Category	Severity	Location	Status
Gas Optimization	🟡 Low	NurseryManagement.sol: 19	Resolved

Description

The property `BZAI` is set in the constructor but never used within the implementation of the contract.

Recommendation

Remove unused properties or pass it to the constructor of the parent class where it is stored.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

NRM-05 | Mapping can be in refactored in struct

Category	Severity	Location	Status
Gas Optimization	🟡 Low	NurseryManagement.sol: 32~36; 32~38	Resolved

Description

These mappings share the same key:

```
32  mapping(uint256 => ZaiStruct.MintedData) public nurseryMintedDatas;
33  mapping(uint256 => ZaiStruct.MintedData) _tempCounter;
34  mapping(uint256 => uint256) _nextStateToMint; // 0 bronze ; 1 Silver ; 2 Gold ; 3 Platinum
35  mapping(uint256 => uint256) public nextUnlock; // use for prevent minting
36  mapping(uint256 => uint256) public lastTimeReserveEgg; // preventing nursery owner reserve one egg per day max
37
38  mapping(uint256 => uint256) public numberOfEggsOffered;
```

Recommendation

Using an optimised struct to store all the informations in 1 mapping would improve gas consumption.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

OCC-01 | LP token can be changed and prevent users from withdrawing

Category	Severity	Location	Status
Volatile Code	● High	OpenAndCloseCenter.sol: 63~65	Resolved

Description

In the contract, `LP` token address can be changed.
If tokens have already be transferred, they would get locked and this would block the contract from functioning correctly.

Recommendation

LP address should be set only once.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change and prevent updating the address of `LP` token.

OCC-02 | Check Effects Interactions pattern Violation

Category	Severity	Location	Status
Volatile Code	● High	OpenAndCloseCenter.sol: 304~310; 331~333; 345~350; 370~372	Resolved

Description

The functions `closeTrainingCenter`, `getBZAIBackFromClosingTraining`, `closeLabo`, `getBZAIBackFromClosingLabo` calls external contracts before updating the local state. ie:

```
require(

ITrainingManagement(gameAddresses.getTrainingCenterAddress())
        .cleanSlotsBeforeClosing(_tokenId)
);

c.isClosing = true;
c.timestampClosedActed = block.timestamp +
closingHousesDuration;
```

Recommendation

Follow the check effects interactions pattern and update the state before external calls.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

OCC-03 | No events emitted

Category	Severity	Location	Status
Language Specific	● Medium	OpenAndCloseCenter.sol	Resolved

Description

The following functions does not emit events to pass informations off chain:

- `setMaturityDuration`
- `setClosingDuration`
- `setGameAddresses`
- `setLpToken`
- `setTrainingCenterPrice`
- `setLaboratoryPrice`

Recommendation

We recommend declaring and emitting corresponding events for all the essential state variables that are possible to be changed during runtime.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

OCC-04 | housesStates should use an enum

Category	Severity	Location	Status
Gas Optimization	● Medium	OpenAndCloseCenter.sol: 28~34	Resolved

Description

The variable `housesStates` uses strings to encode the state resulting in gas consumption overhead.

```
string[5] housesStates = [
    "doesn't_exist",
    "under_construction",
    "open",
    "under_destroyment",
    "destroyed"
];
```

Recommendation

`housesStates` should use an enum to encode the state.

Alleviation

`[UnblockLabs]` The client opted to returns an `uint256` instead of the `string`.

OCC-05 | Mapping can be in refactored in struct

Category	Severity	Location	Status
Gas Optimization	● Low	OpenAndCloseCenter.sol: 40~45; 40~41; 44~45	Resolved

Description

These mappings share the same key:

```
mapping(uint256 => uint256) public trainingCenterMaturityTime;
mapping(uint256 => uint256) public lockedInTrainingCenterID;
mapping(uint256 => uint256) public laboratoryMaturityTime;
mapping(uint256 => uint256) public lockedInLaboID;
```

Recommendation

Using an optimised struct to store all the informations in 1 mapping would improve gas consumption.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

OCC-06 | Addresses from gameAddresses should be cached

Category	Severity	Location	Status
Gas Optimization	🟡 Low	OpenAndCloseCenter.sol	Resolved

Description

Adresses from `gameAddresses` are used in different functions within the implementation of the contract. Each call to `gameAddresses` requires an external call that could be avoided by caching the addresses used.

Recommendation

Cache the addresses returned by `gameAddresses`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

ORA-01 | Weak sources of randomness

Category	Severity	Location	Status
Volatile Code	🟠 High	Oracle1.sol: 48~52	Acknowledge

Description

The parameters used to generate the random number are weak and can be easily guessed by an attacker.

```
48  nonce = uint256(
49      keccak256(
50          abi.encodePacked(_id, block.difficulty, nonce,
    block.number, _balance, block.gaslimit)
51      )
52  );
```

`block.difficulty` is fixed on Polygon (or any PoS networks), `nonce`, `block.number`, and `gas.limit`, are all known prior to the call.
Also `_balance` can be manipulated by sending tokens on the contracts.
This makes it easy for an attacker to simulate the algorithm before sending transactions.

Recommendation

Option 1:
Use an external source of randomness that can't be manipulated. We recommend using Chainlink VRF for all operations that requires a true random number.
We understand that this change can't easily be adapted as it would require a change in the workflow of the fights.

Option 2:
Add more meaningful variables like `block.timestamp`, `tx.origin`, or `msg.sender`, and remove the "fixed" values.

Alleviation

`[UnblockLabs]` The client adapted the algorithm to be deterministic based on the sender and acknowledged that a pseudo number generation is sufficient.

ORA-02 | Parameter _id does not add randomness

Category	Severity	Location	Status
Gas Optimization	● Medium	Oracle1.sol: 36	Resolved

Description

The parameter _id passed to the function getRandom() does not add any randomness to the result returned.

```
function getRandom(bytes32 _id) external returns (uint256)
```

Recommendation

Simplify the code by removing the parameter and doing a computation on the caller contract when required.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

ORA-03 | No functionality added between Oracle1 and Oracle2

Category	Severity	Location	Status
Coding Style	● Low	Oracle1.sol	Resolved

Description

The contracts Oracle1.sol and Oracle2.sol shares an almost identical implementation. Using 2 contracts does not add any randomness to the return values.

Recommendation

We recommend using only 1 contract that generate true random numbers to enhance maintainability.

Alleviation

`[UnblockLabs]` The client opted to use only 1 `Oracle` contract.

PAY-01 | No events emitted

Category	Severity	Location	Status
Coding Style	● Medium	Payments.sol	Resolved

Description

The following functions does not emit events to pass informations off chain:

- `distributeFees`

Recommendation

We recommend declaring and emitting corresponding events for all the essential state variables that are possible to be changed during runtime.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

PAY-02 | Owner and DAO address is the same

Category	Severity	Location	Status
Centralization / Privilege	🟡 Low	Payments.sol	Resolved

Description

The contract `Payments` transfer funds to the owner of the contract. Nothing guarantees this is the DAO.

Recommendation

Define the DAO address as an independent address to separate responsibilities between ownership and rewards.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

PAY-03 | block.timestamp not required in events

Category	Severity	Location	Status
Gas Optimization	● Low	Payments.sol	Resolved

Description

The following events emit block.timestamp as one of their parameters.

- `RewardUsed`
- `RevenuesUsed`
- `RevenuesClaimed`
- `RewardsClaimed`
- `BurnedForEggs`
- `NftOwnerPaid`
- `NftOwnerPaid`
- `RevenuesForOwner`

`block.timestamp` is already included in the event definition.

Recommendation

Remove `block.timestamp` from the events.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

PAY-04 | BZAI property should be immutable

Category	Severity	Location	Status
Gas Optimization	● Low	Payments.sol: 15	Resolved

Description

The property `BZAI` is never changed within the implementation of the contract and should be declared as `immutable`.

Recommendation

Declare the property `BZAI` as `immutable`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

POT-01 | Burn token without approval

Category	Severity	Location	Status
Centralization / Privilege	● Medium	PotionNFT.sol: 249~252	Resolved

Description

The function `burnPotion()` can burn tokens without the approval of the owner.

```
function burnPotion(uint256 _tokenId) external onlyAuth returns
(bool) {
    _burn(_tokenId);
    return true;
}
```

Recommendation

Check for user's approval before burning a potion.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

POT-02 | No events emitted

Category	Severity	Location	Status
Coding Style	● Medium	PotionNFT.sol	Resolved

Description

The following functions does not emit events to pass informations off chain:

- `setXpPotionsPrice`
- `setRestPotionsPrice`
- `setGameAddresses`
- `emptyingPotion`

Recommendation

We recommend declaring and emitting corresponding events for all the essential state variables that are possible to be changed during runtime.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

POT-03 | Missing input validation

Category	Severity	Location	Status
Volatile Code	● Low	PotionNFT.sol: 40~42; 44~46	Resolved

Description

Price can be set to **0** in the functions `setXpPotionsPrice`, `setRestPotionsPrice`.

Recommendation

Validate the function's input parameters to ensure a minimum price.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

POT-04 | Gas optimisation in offerPotion() and mintPotionForSale()

Category	Severity	Location	Status
Gas Optimization	🟡 Low	PotionNFT.sol: 64~81	Resolved

Description

The functions `offerPotion()` and `mintPotionForSale()` could improve the gas consumption by using `if; else;` statements as only 1 condition should evaluate to `true`.

```
if (_type == 0) {
    i.powers.water = _power;
}
if (_type == 1) {
    i.powers.fire = _power;
}
if (_type == 2) {
    i.powers.metal = _power;
}
if (_type == 3) {
    i.powers.air = _power;
}
if (_type == 4) {
    i.powers.stone = _power;
}
if (_type == 8) {
    i.powers.mana = _power * 100;
}
```

Recommendation

Use `if; else;` statements to improve gas consumption.

Alleviation

`[UnblockLabs]` The client opted to refactor the code code and use `if; else;` statements.

RCR-01 | Mix of responsibility

Category	Severity	Location	Status
Volatile Code	🟠 High	ReserveForChallengeRewards.sol: 24~37	Resolved

Description

The function `updateRewards()` does not verify that it is called only 1 time per day. Instead, the contract expect the caller to limit the calls. This can result in more than `daylyAddOn` transferred during a day.

Recommendation

Check within the contract's implementation that maximum `daylyAddOn` is transferred in 1 day.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

RCR-02 | Gas optimisation in updateRewards()

Category	Severity	Location	Status
Gas Optimization	🟡 Low	ReserveForChallengeRewards.sol: 29~30	Resolved

Description

The function `balanceOf` is called twice for the same account in the function `updateRewards()`.

Recommendation

Store the result in a local variable and call the function only once.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

RCR-03 | Invalid comment

Category	Severity	Location	Status
Volatile Code	● Information	ReserveForChallengeRewards.sol: 7; 13	Resolved

Description

The description mentions 50 months whereas the math are done with 5 years (= 60 months).

```
// some rewards have 50 Months unlocking period
```

```
uint256 public daylyAddOn = 27397 * 1E18; //   50M / (365 * 5)
```

Recommendation

Update the code or the comment to match.

Alleviation

`[UnblockLabs]` The client opted to update the comment.

RWR-01 | Mix of responsibility

Category	Severity	Location	Status
Volatile Code	● High	ReserveForWinRewards.sol	Resolved

Description

The function `updateRewards()` does not verify that it is called only 1 time per hour. Instead, the contract expect the caller to limit the calls. This can result in more than `hourlyAddOn` transferred during an hour.

Recommendation

Check within the contract's implementation that maximum `hourlyAddOn` is transferred in 1 hour.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

RWR-02 | Gas optimisation in updateRewards()

Category	Severity	Location	Status
Gas Optimization	🟡 Low	ReserveForWinRewards.sol: 31~32	Resolved

Description

The function `balanceOf` is called twice for the same account in the function `updateRewards()`.

Recommendation

Store the result in a local variable and call the function only once.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

RWR-03 | BZAI property should be immutable

Category	Severity	Location	Status
Gas Optimization	● Low	ReserveForWinRewards.sol: 13	Resolved

Description

The property `BZAI` is never changed within the implementation of the contract and should be declared as `immutable`.

Recommendation

Declare the property `BZAI` as `immutable`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

RPP-01 | No events emitted

Category	Severity	Location	Status
Coding Style	● Medium	RewardsPvP.sol	Resolved

Description

The following functions does not emit events to pass informations off chain:

- `setRewardPortion`
- `setGameAddresses`

Recommendation

We recommend declaring and emitting corresponding events for all the essential state variables that are possible to be changed during runtime.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

RPP-02 | ReentrancyGuard not required

Category	Severity	Location	Status
Gas Optimization	🟡 Low	RewardsPvP.sol: 41~51	Resolved

Description

The function `getWinningRewards` uses the modifier `nonRentrant` though it is not required here.

Recommendation

Remove `ReentrancyGuard` dependency from the contract.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

RPP-03 | Addresses from gameAddresses should be cached

Category	Severity	Location	Status
Gas Optimization	🟡 Low	RewardsPvP.sol: 24; 49	Resolved

Description

Adresses from `gameAddresses` are used in different functions within the implementation of the contract. Each call to `gameAddresses` requires an external call that could be avoided by caching the addresses used.

Recommendation

Cache the addresses returned by `gameAddresses`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

RPP-04 | BZAI property should be immutable

Category	Severity	Location	Status
Gas Optimization	🟡 Low	RewardsPvP.sol: 15	Resolved

Description

The property `BZAI` is never changed within the implementation of the contract and should be declared as `immutable`.

Recommendation

Declare the property `BZAI` as `immutable`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

RRF-01 | Mix of responsibility

Category	Severity	Location	Status
Volatile Code	● High	RewardsRankingFound.sol: 74~88; 90~100	Resolved

Description

The functions `getDailyRewards()` and `getWeeklyRewards()` does not verify that it is called only 1 time per period. Instead, the contract expect the caller to limit the calls. This can result in more token transferred during a period.

Recommendation

Check within the contract's implementation that the rewards to not exceed the limit for the period.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

RRF-02 | setGameAddresses() should call updateAddresses()

Category	Severity	Location	Status
Coding Style	● Medium	RewardsRankingFound.sol: 37~40	Acknowledge

Description

After updating `gameAddresses`, the linked cached addresses are not automatically reloaded. This could lead to unexpected behaviors of the contract.

```
function setGameAddresses(address _address) external onlyOwner {
    require(gameAddresses == IAddresses(address(0x0)), "Already setted");
    gameAddresses = IAddresses(_address);
}
```

Recommendation

Call `updateAddresses()` inside the function `setGameAddresses()`.

Alleviation

`[UnblockLabs]` The client implemented a method `updateInterfaces()`, called at the end of the deployment of the contracts.

.

RRF-03 | No events emitted

Category	Severity	Location	Status
Coding Style	● Medium	RewardsRankingFound.sol	Resolved

Description

The following functions does not emit events to pass informations off chain:

- `setGameAddresses`
- `updateAddresses`
- `balancerToPvpReward`
- `balancerToWinPveReward`
- `getDailyRewards`

Recommendation

We recommend declaring and emitting corresponding events for all the essential state variables that are possible to be changed during runtime.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

RRF-04 | Use an amount as a parameter in balancerToPvpReward() and balancerToWinPveReward()

Category	Severity	Location	Status
Coding Style	🟡 Low	RewardsRankingFound.sol: 50~54; 56~60	Resolved

Description

The functions `balancerToPvpReward()` and `balancerToWinPveReward()` can be called by the owner without restrictions, after the unlock timestamp.
The algorithm transfer 10% of the available tokens to the rewards contracts. Taking an amount as a parameter with a limit per period could help reach the desired balance more effectively.

Recommendation

Take the amount to transfer as a parameter of the functions.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

RRF-05 | Properties should be immutable

Category	Severity	Location	Status
Gas Optimization	🟡 Low	RewardsRankingFound.sol: 13; 19	Resolved

Description

The properties `unlockBalancerTimestamp` and `IReserve` are never changed within the implementation of the contract and should be declared as `immutable`.

Recommendation

Declare the properties as `immutable`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

RRF-06 | Use of Reentrancy guard

Category	Severity	Location	Status
Gas Optimization	🟡 Low	RewardsRankingFound.sol: 74~78; 90~94	Resolved

Description

The functions `getDailyRewards()` and `getWeeklyRewards()` use the modifier `nonReentrant`, but do not really requires it as the `BZAI` token is controlled by the project and the function can only be called by internal contracts.

Recommendation

Remove the `nonReentrant` modifier and the dependency to `ReentrancyGuard.sol` if not used.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

RWT-01 | No events emitted

Category	Severity	Location	Status
Coding Style	● Medium	RewardsTournament.sol: 19~29	Resolved

Description

The following functions does not emit events to pass informations off chain:

- `getRewardsForTournament()`

Recommendation

We recommend declaring and emitting corresponding events for all the essential state variables that are possible to be changed during runtime.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

RWT-02 | getRewardsForTournament() transfers tokens to the owner account

Category	Severity	Location	Status
Centralization / Privilege	🟡 Low	RewardsTournament.sol: 28	Acknowledge

Description

The functions getRewardsForTournament() transfers the claimable tokens to the owner account.

```
require(BZAI.transfer(msg.sender, futurPartClaimaible));
```

The usage of the funds can't be tracked by the community this way.

Recommendation

Use a specific address to receive the rewards and communicate publicly on how the funds are used.

Alleviation

`[UnblockLabs]` The client opted to keep the implementation as is.

RWT-03 | Potential transfer of 0 tokens

Category	Severity	Location	Status
Volatile Code	● Low	RewardsTournament.sol: 28	Resolved

Description

The function `getRewardsForTournament()` does not validate that the amount to transfer is greater than **0**.

```
require(BZAI.transfer(msg.sender, futurPartClaimaible));
```

Recommendation

Add a test to execute the transfer only when `futurPartClaimaible > 0`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

RWT-04 | BZAI property should be immutable

Category	Severity	Location	Status
Gas Optimization	🟡 Low	RewardsRankingFound.sol: 11	Resolved

Description

The property `BZAI` is never changed within the implementation of the contract and should be declared as `immutable`.

Recommendation

Declare the property `BZAI` as `immutable`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

RWF-01 | setHourlyBlockQuantity can change reward emission

Category	Severity	Location	Status
Centralization / Privilege	● Medium	RewardsWinningFound.sol: 72~74	Resolved

Description

The function setHourlyBlockQuantity() can be used by the owner to reduce or increase the period between collection of rewards. By setting a high value, the rewards won't get transferred to the contract and can potentially prevent the users from getting rewards.

Recommendation

Validate the range of acceptable values for `hourlyBlockQuantity` in `setHourlyBlockQuantity()`, or remove the capacity to edit it if not required.

Alleviation

`[UnblockLabs]` The client opted to change the implementation to not rely on a fixed number of block per hour.

RWF-02 | No events emitted

Category	Severity	Location	Status
Coding Style	● Medium	RewardsWinningFound.sol	Resolved

Description

The following functions does not emit events to pass informations off chain:

- `setGameAddresses`
- `updateAddresses`
- `balancerToPvpReward`
- `balancerToRankingReward`
- `setHourlyBlockQuantity`
- `setRewardPortion`
- `setBonusMult`

Recommendation

We recommend declaring and emitting corresponding events for all the essential state variables that are possible to be changed during runtime.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

RWF-03 | Low resolution for bonusMult

Category	Severity	Location	Status
Coding Style	● Medium	RewardsWinningFound.sol: 85; 116	Resolved

Description

The property `bonusMult` can only be set to the values **10**, **11**, **12** which will result in a bonus of 0, 10% or 20%.

```
require(_bonus >= 10 && _bonus <= 12, "bonus multiplicator not match");
```

The resolution used later to affect the bonus does not allow intermediate values (between 0 and 20%).

```
_toSend = _toSend * bonusMult / 10;
```

Recommendation

Increase the resolution.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

RWF-04 | Use an amount as a parameter in balancerToPvpReward() and balancerToWinPveReward()

Category	Severity	Location	Status
Coding Style	🟡 Low	RewardsWinningFound.sol	Resolved

Description

Description

The functions `balancerToPvpReward()` and `balancerToRankingReward()` can be called by the owner without restrictions, after the unlock timestamp.
The algorithm transfer 10% of the available tokens to the rewards contracts. Taking an amount as a parameter with a limit per period could help reach the desired balance more effectively.

Recommendation

Take the amount to transfer as a parameter of the functions.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

RWF-05 | Properties should be immutable

Category	Severity	Location	Status
Gas Optimization	🟡 Low	RewardsWinningFound.sol: 14~15	Resolved

Description

The properties `BZAI` and `IReserve` are never changed within the implementation of the contract and should be declared as `immutable`.

Recommendation

Declare the properties as `immutable`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

TRC-01 | Burn without approval of the owner

Category	Severity	Location	Status
Centralization / Privilege	● High	TrainingCenter.sol: 114~116	Resolved

Description

The function `burn()` allows the game to burn tokens without verifying the user's approval.

```
function burn(uint256 _tokenId) external onlyAuth {
    _burn(_tokenId);
}
```

Recommendation

Verify the owner's approval before burning tokens.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

TRC-02 | Check Effects Interactions pattern Violation

Category	Severity	Location	Status
Volatile Code	● High	TrainingCenter.sol: 87~88	Resolved

Description

In the function `_mintTrainingCenter()`, the state is updated after the mint, allowing a potential reentrancy attack.

```
_safeMint(_to, _newItemId);
_numberOfTrainingSpots[_newItemId] = 3;
```

Recommendation

Move the call to `_safemint()` after the state update.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

TRC-03 | No restriction on _preMint in the constructor

Category	Severity	Location	Status
Volatile Code	● Medium	TrainingCenter.sol: 27~32	Acknowledge

Description

The parameter `_preMint` is not limited and the deployment can overflow the block gas limit if the value is too high.

Recommendation

Create an independent function out of the constructor to pre-mint items and pass the quantity as a parameter of the function.

Alleviation

`[UnblockLabs]` The client opted to keep the implementation as is.
`[BandZai]` PreMint Number will be 25 and won't failed.

TRC-04 | No events emitted

Category	Severity	Location	Status
Coding Style	● Medium	TrainingCenter.sol	Resolved

Description

The following functions does not emit events to pass informations off chain:

- `setCID`
- `setGameAddresses`

Recommendation

We recommend declaring and emitting corresponding events for all the essential state variables that are possible to be changed during runtime.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

TRC-05 | setGameAddresses should be external

Category	Severity	Location	Status
Gas Optimization	🟡 Low	TrainingCenter.sol: 71~74	Resolved

Description

The function `setGameAddresses()` is public but not called within the contract's implementation.

Recommendation

Set the function's visibility to `external`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

TRC-06 | Addresses from gameAddresses should be cached

Category	Severity	Location	Status
Gas Optimization	🟡 Low	TrainingCenter.sol	Resolved

Description

Adresses from `gameAddresses` are used in different functions within the implementation of the contract. Each call to `gameAddresses` requires an external call that could be avoided by caching the addresses used.

Recommendation

Cache the addresses returned by `gameAddresses`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

TRM-01 | Missing input validation

Category	Severity	Location	Status
Volatile Code	● High	TrainingManagement.sol: 191~230	Resolved

Description

The function setTrainingSpot() does not validate the `_coachPercentPayment` passed as parameter, potentially allowing a payment of more than **100%**.

Recommendation

Add a test to verify that the parameters are within an accepted range.

Alleviation

`[UnblockLabs]` The client opted to implementation a limitation to 90%.

TRM-02 | Potential underflow in _updateZai

Category	Severity	Location	Status
Volatile Code	● High	TrainingManagement.sol: 439	Resolved

Description

The function _updateZai() will underflow if the `level` of the Zai is greater than the level of the coach.

```
(c.level - z.level > 0)
```

Recommendation

Update the test to never underflow.
ie:

```
(c.level > z.level)
```

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

TRM-03 | Check Effects Interactions pattern Violation

Category	Severity	Location	Status
Volatile Code	● High	TrainingManagement.sol: 254~257	Resolved

Description

In the function `registerCoaching()`, the state is updated after doing an external call.

```
IZai.updateStatus(_zaiId, 2, _trainingId);
t.coach.coachId = _zaiId;
t.spotOpened = true;
t.coach.currentCoachLevel = z.level;
```

Recommendation

Update the local state before doing any external calls.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

TRM-04 | Differences between cleanSpot and kickCoachFromSpot

Category	Severity	Location	Status
Volatile Code	● Medium	TrainingManagement.sol: 271~281; 283~289	Resolved

Description

The owner of a training center can call the method `cleanSpot()` to bypass the checks done in `kickCoachFromSpot()`, and potentially remove a coach before the end of the training.

Recommendation

Move the validation to `cleanSpot()` and remove the function `kickCoachFromSpot()`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change and removed the function `kickCoachFromSpot()`.

TRM-05 | No events emitted

Category	Severity	Location	Status
Coding Style	● Medium	TrainingManagement.sol	Resolved

Description

The following functions does not emit events to pass informations off chain:

- `setminimumTrainingPrice`
- `setGameAddresses`

Recommendation

We recommend declaring and emitting corresponding events for all the essential state variables that are possible to be changed during runtime.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

TRM-06 | slotStatus should be an enum

Category	Severity	Location	Status
Gas Optimization	● Medium	TrainingManagement.sol: 38~45	Resolved

Description

The variable slotStatus uses string to describe the possible states.

```
string[6] slotStatus = [
    "not_set",
    "closed",
    "free",
    "in_use",
    "waiting_coach",
    "waiting_for_finish_training"
];
```

This variable can be represented by a numeric value in an Enum to improve gas consumption.

Recommendation

Use an enum instead of a collection of strings.
ie:

```
enum SlotStatus {
    NotSet,
    Closed,
    Free,
    InUse,
    WaitingCoach,
    WaitingForFinishTraining
};
```

Alleviation

`[UnblockLabs]` The client opted to update the implementation to return `uint256` instead of a `string`.

TRM-07 | Event TrainingPurchase should emit more informations

Category	Severity	Location	Status
Coding Style	● Low	TrainingManagement.sol: 23	Resolved

Description

The event `TrainingPurchase` does not emit all the informations required to recreate the state off-chain.

```
event TrainingPurchase(address indexed trainingOwner, uint256
price);
```

Recommendation

The following informations should be passed off-chain: `buyer address`, `trainingId`, `zaiId`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

TRM-08 | Event CoachPaid should emit more informations

Category	Severity	Location	Status
Coding Style	🟡 Low	TrainingManagement.sol: 24	Resolved

Description

The event `CoachPaid` does not emit all the informations required to recreate the state off-chain.

```
event CoachPaid(address indexed coachOwner, uint256 price);
```

Recommendation

The following informations should be passed off-chain: `buyer address`, `trainingId`, `coachId`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

TRM-09 | Unused property

Category	Severity	Location	Status
Gas Optimization	🟡 Low	TrainingManagement.sol: 14	Resolved

Description

The property `BZAI` is set in the constructor but never used within the implementation of the contract.

Recommendation

Remove unused properties or pass it to the constructor of the parent class where it is stored.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

TRM-10 | maxDurationTraining should be constant

Category	Severity	Location	Status
Gas Optimization	🟡 Low	TrainingManagement.sol: 29	Resolved

Description

The property `maxDurationTraining` is never changed within the implementation of the contract and should be declared as a constant.

Recommendation

Declare `maxDurationTraining` as a constant.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

TRM-11 | setGameAddresses should be external

Category	Severity	Location	Status
Gas Optimization	🟡 Low	TrainingManagement.sol: 100~103	Resolved

Description

The function `setGameAddresses()` is public but not called within the contract's implementation.

Recommendation

Set the function's visibility to `external`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

TRM-12 | coachDatas should be in CapWords style

Category	Severity	Location	Status
Coding Style	● Information	TrainingManagement.sol: 47~53	Resolved

Description

To follow the coding patterns, structures should be in CapWords style.

Recommendation

Change the naming to `CoachDatas`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

ZFT-01 | Missing validation of reward repartition

Category	Severity	Location	Status
Volatile Code	● High	ZaiFighting.sol: 400~451; 422~440	Resolved

Description

The function `_paySchoolarAndOwner()` does not verify that the total of the percentages assigned is not above **100%**. If the platform fee is activated, the fee should be deducted first from the total rewards.

```
422  _scholarReward =
423      (_reward * _scholarDatas.guildeDatas.percentageForScholar) /
424      100;
425  _ownerReward =
426      (_reward * _scholarDatas.guildeDatas.percentageForGuilde) /
427      100;
428
429  _scholarAddress = _scholarDatas.guildeDatas.renterOf;
430  _ownerAddress = _scholarDatas.guildeDatas.masterOf;
431
432  require(
433      IPay.rewardPlayer(
434          _scholarDatas.guildeDatas.platformAddress,
435          (_reward *
436              _scholarDatas.guildeDatas.percentagePlatformFees) /
     100,
437          0,
438          0
439      )
440  );
```

Recommendation

Verify that the sum of percentage do not exceed 100%. Deduct all fees prior to the repartition.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

ZFT-02 | Rounding error in _paySchoolarAndOwner()

Category	Severity	Location	Status
Volatile Code	● Medium	ZaiFighting.sol: 412~418; 416~418; 425~427	Resolved

Description

The function `_paySchoolarAndOwner()` does not account for rounding when applying the different percentages.

```
_ownerReward =
    (_reward *
        (100 - _scholarDatas.delegateDatas.percentageForScholar)) /
    100;
_scholarReward =
    (_reward * _scholarDatas.delegateDatas.percentageForScholar) /
    100;
```

Recommendation

Use a subtraction for the last percentage.
ie:

```
_ownerReward =
    (_reward *
        (100 - _scholarDatas.delegateDatas.percentageForScholar)) /
    100;
_scholarReward = _reward - _ownerReward;
```

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

ZFT-03 | No events emitted

Category	Severity	Location	Status
Coding Style	● Medium	ZaiFighting.sol	Resolved

Description

The following functions does not emit events to pass informations off chain:

- `setGameAddresses`
- `pauseUnpauseGame`
- `setXpRewardByFight`
- `setBzaiRewardCountPerDay`
- `setRegenerationDuration`
- `useRestPotion`

Recommendation

We recommend declaring and emitting corresponding events for all the essential state variables that are possible to be changed during runtime.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

ZFT-04 | Multiplication on the result of a division

Category	Severity	Location	Status
Language Specific	● Low	ZaiFighting.sol: 316	Resolved

Description

The function `_getXpToWin()` performs a multiplication on the result of a division.

```
_xp = ((2 * _xp) - ((_xp / _totalPowers) * _totalUsedPowers)) /
100;
```

Since the types. are uint, this can lead to a lost of precision and rounding errors.

Recommendation

Avoid performing a multiplication on the result of a division. The code can be updated to produce the same result, ie:

```
_xp = ((2 * _xp) - (_xp * _totalUsedPowers / _totalPowers)) / 100;
```

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

ZFT-05 | Event FightResult should emit more informations

Category	Severity	Location	Status
Coding Style	🟡 Low	ZaiFighting.sol: 32~36	Resolved

Description

The event `FightResult` does not emit all the informations required to recreate the state off-chain.

Recommendation

We recommend passing the address of the player to the event.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

ZFT-06 | Addresses from gameAddresses should be cached

Category	Severity	Location	Status
Gas Optimization	🟡 Low	ZaiFighting.sol	Resolved

Description

Adresses from `gameAddresses` are used in different functions within the implementation of the contract. Each call to `gameAddresses` requires an external call that could be avoided by caching the addresses used.

Recommendation

Cache the addresses returned by `gameAddresses`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

ZFT-07 | Address of oracle contract should be cached

Category	Severity	Location	Status
Gas Optimization	🟡 Low	ZaiFighting.sol: 497	Resolved

Description

The function `_generateRandomDatas()` execute an external call on every calls to load the address of the oracle contract. As noted in BZA-03, only 1 oracle should be used. `ZaiFighting` should cache this address to improve gas consumption.

Recommendation

Use only 1 oracle and cache the address.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

ZFT-08 | If; statement not required

Category	Severity	Location	Status
Gas Optimization	🟡 Low	ZaiFighting.sol: 466~492	Resolved

Description

The function `_getZaiPowersByElement()` executes a `if;` statement before running the loop. Since the loop will not run if the condition evaluate to `false`, the `if;` statement can be removed.

```
if (_potions.length > 0) {
  for (uint256 i = 0; i < _potions.length; i++) {
      ...
  }
}
```

Recommendation

Remove the `if;` statement.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

ZFL-01 | Mix of index and power values

Category	Severity	Location	Status
Volatile Code	● High	ZaiFightingLibrary.sol: 113~119	Resolved

Description

The function `_getPattern()` mistakenly assign the index of the loop instead of the power value.

```
for(uint256 i = 0 ; i < 5;){
    if(_powers[i] > 0){
        _activePowers[activeIndex] = i;
        unchecked {++activeIndex;}
    }
    unchecked {++i;}
}
```

Recommendation

Replace the assignation to _activePowers to:

```
_activePowers[activeIndex] = _powers[i];
```

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

ZFL-02 | Gas optimisation in updateFightingProgress()

Category	Severity	Location	Status
Gas Optimization	● Medium	ZaiFightingLibrary.sol: 11~15	Resolved

Description

The function `updateFightingProgress()` calls `_winTheRound()` 3 times with the same parameters and the function returns the same results all the time (win / draw / loose).

```
if(_winTheRound(_elements[i],_toReturn[i+3]) == 1){
    _toReturn[1] += _powers[i]; // My score
}else if(_winTheRound(_elements[i],_toReturn[i+3]) == 0){
    _toReturn[2] += _toReturn[i+12]; //challenger score
}else if(_winTheRound(_elements[i],_toReturn[i+3]) == 2){
```

Recommendation

Add a local variable to store the result of `_winTheRound()` and reuse it within the function.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

ZFL-03 | Gas optimisation in getUsedPowersByElement()

Category	Severity	Location	Status
Gas Optimization	🟡 Low	ZaiFightingLibrary.sol: 239~244	Resolved

Description

The function `getUsedPowersByElement()` can improve gas consumption by using `if; else;` statements.

```
239  if(_powers[i] == 0){
240      require(_elements[i] == 5, "Cheat!");
241  }
242  if(_powers[i] > 0 && _elements[i] != 5){
243      usedPowers[_elements[i]] += _powers[i];
244  }
```

Recommendation

Use `if; else;` statements.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

ZMT-01 | Missing address(0) validation

Category	Severity	Location	Status
Volatile Code	● Medium	ZaiMeta.sol: 27~31	Resolved

Description

The value of the parameter `_levelStorage` is not validated in the constructor of the contract.

```
constructor(string[7] memory _names, address _levelStorage) {
    _godNames[1] = _names;
    levelStorage = _levelStorage;
    ILevel = ILevelStorage(_levelStorage);
}
```

This parameter can't be changed once deployed and can result in an invalid contract.

Recommendation

Validate that `_levelStorage` is not equal to `address(0)`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

ZMT-02 | No events emitted

Category	Severity	Location	Status
Coding Style	● Medium	ZaiMeta.sol	Resolved

Description

The following functions does not emit events to pass informations off chain:

- `setGameV2optionsAddress`
- `setGameAddresses`
- `setGodNames`
- `updateStatus`
- `updateMana`

Recommendation

We recommend declaring and emitting corresponding events for all the essential state variables that are possible to be changed during runtime.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

ZMT-03 | Gas optimisation in _createZaiDatas()

Category	Severity	Location	Status
Gas Optimization	🟠 Medium	ZaiMeta.sol: 225~226	Resolved

Description

In the function `_createZaiDatas()`, the following 2 lines of codes are only used in the `else;` branch of the algorithm.

```
225 uint256 random = _getRandom(_to, _ipfsId);
226 uint256 _points = (_level * 3) + 8;
```

Recommendation

Move the code inside the `else;` branch to only execute it when required and reduce gas consumption.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

ZMT-04 | Properties should be constant

Category	Severity	Location	Status
Gas Optimization	● Low	ZaiMeta.sol: 17~19	Resolved

Description

The following properties are never changed within the implementation of the contract.

```
uint256 fivePowersMinLevel = 15;
uint256 fourPowersMinLevel = 10;
uint256 threePowersMinLevel = 5;
```

Recommendation

Declare the properties as `constant`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

ZMT-05 | Duplicate variables

Category	Severity	Location	Status
Gas Optimization	🟡 Low	ZaiMeta.sol: 12; 14	Resolved

Description

The variables `ILevelStorage` and `levelStorage` stores the same information.

Recommendation

Remove `levelStorage` and use `address(ILevelStorage)` when required.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

ZMT-06 | Unnecessary loop

Category	Severity	Location	Status
Gas Optimization	🟡 Low	ZaiMeta.sol: 253~259	Resolved

Description

The function `updateXp()` execute an unnecessary loop to create the challengers.

```
for (uint256 i = 0; i < 3; ) {
    uint256 _newItemId = IZai.createNewChallenger();
    _preMintZai(level, _newItemId);
    unchecked {
        ++i;
    }
}
```

Recommendation

Remove the loop and call `_preMintZai()` directly.
ie:

```
_preMintZai(level, IZai.createNewChallenger());
_preMintZai(level, IZai.createNewChallenger());
_preMintZai(level, IZai.createNewChallenger());
```

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

ZNF-01 | Potential loss of "piggybank"

Category	Severity	Location	Status
Volatile Code	● High	ZaiNFT.sol: 110~117	Resolved

Description

The function `burnZai()` can be called directly by the owner of the `Zai`. In that case, the user will loose the "piggybank" rewards associated with his token in the `Payment` contract.

Recommendation

Restrict the function to be called only by the `burnZaiToGetHisPiggyBank()` function in `Payment` contract.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

ZNF-02 | Missing address(0) validation

Category	Severity	Location	Status
Volatile Code	● Medium	ZaiNFT.sol: 15~21	Resolved

Description

The value of the parameter `_levelStorage` is not validated in the constructor of the contract.

```
constructor(address _zaiMeta, address _levelStorage)
    ERC721("BandZai_NFT_ZAI", "ZAI")
{
    zaiMeta = IZaiMeta(_zaiMeta);
    zaiMetaAddress = _zaiMeta;
    levelStorage = _levelStorage;
}
```

This parameter can't be changed once deployed and can result in an invalid contract.

Recommendation

Validate that `_levelStorage` is not equal to `address(0)`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

ZNF-03 | Duplicate functionality from base class

Category	Severity	Location	Status
Gas Optimization	🟠 Medium	ZaiNFT.sol: 12~13	Resolved

Description

The counter `_tokenIds` duplicates the functionality already present in the base class `ERC721Enumerable` which already exposes `totalSupply()`. The value of `totalSupply` is always identical to `_tokenIds.current()`.

Recommendation

Remove the `_tokenIds` variable and use `totalSupply()` from base class.

Alleviation

`[UnblockLabs]` The client opted to change the implementation to inherit directly `ERC721`.

ZNF-04 | Shadows of existing variable name

Category	Severity	Location	Status
Language Specific	🟡 Low	ZaiNFT.sol: 65	Resolved

Description

The function `mintZai()` uses `_name` as a parameter which overrides `_name` variable from `ERC721Enumerable.`

Recommendation

Change the `_name` parameter to `name_`.

Alleviation

`[UnblockLabs]` The client opted to make the recommended change and renamed the parameter to `_zaiName`.

ZST-01 | Invalid event emitted

Category	Severity	Location	Status
Volatile Code	● High	ZaiStats.sol: 140~152	Resolved

Description

In the function `updateCounterWinLoss()`, the wrong event is emitted when a new loss king is set.

```
else if (_fightProgress[2] > _fightProgress[1]) {
            p1.zaiTotalLoss += 1;
            if (p1.zaiTotalLoss > lossKing.totalScore) {
                emit newDrawKing(
                    _zaiId,
                    lossKing.actualKing,
                    block.timestamp - lossKing.kingSince
                );
                lossKing.totalScore = p1.zaiTotalLoss;
                lossKing.kingSince = block.timestamp;
                lossKing.actualKing = _zaiId;
            }
        }
```

Recommendation

Replace the event with `newLossKing`.

Alleviation

`[UnblockLabs]` The client opted to remove this contract from the protocol.

ZST-02 | Events emitted before state change

Category	Severity	Location	Status
Coding Style	● Medium	ZaiStats.sol: 92~99; 105~112; 119~126; 131~138; 143~150	Resolved

Description

In the function `updateCounterWinLoss()`, events are emitted before updating the state on-chain.

Recommendation

Follow the check effects interaction pattern and emit the events after updating the state.

Alleviation

`[UnblockLabs]` The client opted to remove this contract from the protocol.

ZST-03 | Unused properties on-chain

Category	Severity	Location	Status
Gas Optimization	● Medium	ZaiStats.sol: 30~31	Resolved

Description

The properties `_totalDayFight` and `_totalWeekFight` are only set within the implementation of the contract, and never used.

Recommendation

Remove unused properties and recreate them off-chain by monitoring events in the contract if required.

Alleviation

`[UnblockLabs]` The client opted to remove this contract from the protocol.

ZST-04 | Gas optimisation in updateCounterWinLoss()

Category	Severity	Location	Status
Gas Optimization	● Medium	ZaiStats.sol: 82	Resolved

Description

The function `updateCounterWinLoss()` takes an array of **30** `uint256` elements as parameter but uses only 2 of the values to calculate the status win / loss / draw.

Recommendation

Simplify the call to the function by passing a flag indicating the result of the fight to improve gas consumption.

Alleviation

`[UnblockLabs]` The client opted to remove this contract from the protocol.

ZST-05 | Unused import

Category	Severity	Location	Status
Gas Optimization	🟡 Low	ZaiStats.sol: 4	Resolved

Description

The contracts `ZaiStats` import IERC20 but never uses it.

Recommendation

Remove unused imports.

Alleviation

`[UnblockLabs]` The client opted to remove this contract from the protocol.

ZST-06 | Addresses from gameAddresses should be cached

Category	Severity	Location	Status
Gas Optimization	🟡 Low	ZaiStats.sol	Resolved

Description

Adresses from `gameAddresses` are used in different functions within the implementation of the contract. Each call to `gameAddresses` requires an external call that could be avoided by caching the addresses used.

Recommendation

Cache the addresses returned by `gameAddresses`.

Alleviation

`[UnblockLabs]` The client opted to remove this contract from the protocol.

ZST-07 | Events should be named using the CapWords style

Category	Severity	Location	Status
Coding Style	● Information	ZaiStats.sol: 23~26	Resolved

Description

To follow the naming conventions, event names should be named using the CapWords style.

```
event newWinKing(uint256 newKing, uint256 lastKing, uint256 during);
event newDrawKing(uint256 newKing, uint256 lastKing, uint256 during);
event newLossKing(uint256 newKing, uint256 lastKing, uint256 during);
event newTotalKing(uint256 newKing, uint256 lastKing, uint256 during);
```

Recommendation

Rename the events with an uppercase letter at the beginning.

Alleviation

`[UnblockLabs]` The client opted to remove this contract from the protocol.

GLB-01 | Centralization related risks

Category	Severity	Location	Status
Centralization / Privilege	● High		Acknowledge

Description

Any compromise to the owner’s private key account may allow an attacker to take advantage of his authority and mint new tokens, manipulate the parameters and rewards of the game, or block the users from playing.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralisation, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralised privileges or roles in the protocol be improved via a decentralised mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign combination mitigate by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;

AND

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;

AND

- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, mitigate by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;

AND

- Introduction of a DAO/governance/voting module to increase transparency and user involvement;

AND

- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered fully resolved.

- Renounce the ownership and never claim back the privileged roles;

OR

- Remove the risky functionality.

Alleviation

`[UnblockLabs]` The client acknowledged this issue and will work to improve security and transparency around privilege acgtions.

GLB-02 | Anti bot prevention

Category	Severity	Location	Status
Coding Style	● High		Mitigated

Description

The protocol does not implement any controls to prevent smart contracts or bots to play the game. For example, all actions involving randomness can be played from an external contract. This would allow to revert transaction when needed and only execute winning games.

Recommendation

Implement anti-bot mesures to prevent smart contracts from playing and simulating the results of the game before executing a transaction.

Alleviation

`[UnblockLabs]` The client implemented a restriction on the `tx.origin` to try to prevent smart contracts from playing the game.
Though this solution can't guarantee that an automated player won't play the game, it should prevent abuse of the random functions in most cases.

GLB-03 | Use of ERC721Enumerable

Category	Severity	Location	Status
Gas Optimization	● Medium		Resolved

Description

The different NFT contracts inherits from `ERC721Enumerable` and most of the time don't use its functionalities on-chain.
This contract is quite gas consuming and most of its functionalities can be recreated off chain by listening to events emitted by the contract.

Recommendation

If the functionalities provided by `ERC721Enumerable` are not used on-chain, we recommend using the “classic” implementation of `ERC721`.

Alleviation

`[UnblockLabs]` The client updated most of the NFT contracts to inherit from `ERC721`.

GLB-04 | Coding practice

Category	Severity	Location	Status
Coding Style	● Information		Resolved

Description

To follow the naming conventions:

- Constant should be uppercase
- Properties and function names should use mixed casing
- Properties visibility should be explicit
- Properties should be declared before the constructor
- Functions should be declared after the constructor
- Properties and variables should be initialised

Alleviation

`[UnblockLabs]` The client opted to make the recommended change and adapted the code to follow most of the coding practices highlighted above.

GLB-05 | Interface inheritance

Category	Severity	Location	Status
Coding Style	● Information		Acknowledge

Description

The contracts does not inherit from their corresponding interface class, ie: `ChickenNFT` does not inherit `IChicken`.
This can lead to discrepancies between the class and the interfaces without being notified by the compiler.

Recommendation

We recommend inheriting interfaces in the concrete class when possible.

Alleviation

`[UnblockLabs]` The client acknowledged the recommendation.

GLB-06 | Do not cast address(0) to an interface

Category	Severity	Location	Status
Coding Style	● Information		Resolved

Description

The following test is not valid as address(0) won't implement the interface.

```
require(
    gameAddresses == IAddresses(address(0x0)),
    "game addresses already setted"
);
```

Recommendation

Cast the interface to an address and compare it to address(0).
ie:

```
require(
    address(gameAddresses) == address(0x0),
    "game addresses already setted"
);
```

Alleviation

`[UnblockLabs]` The client opted to make the recommended change.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum calculation method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure

Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file.

The result is hexadecimal encoded and is the same as the output of the Linux `sha256sum` command against the target file.

Privileges

The Payments and LiquidityMining contracts on which BZAI and LP tokens can be temporarily stored do not have administration functions allowing transfers/withdrawals. Thus the contract administrator has no control over the tokens of the users.

Disclaimer

and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED “AS IS” AND “AS AVAILABLE” AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, UNBLOCK LABS HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, UNBLOCK LABS SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, UNBLOCK LABS MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER’S OR ANY OTHER PERSON’S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, UNBLOCK LABS PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER’S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER UNBLOCK LABS NOR ANY OF UNBLOCK LABS’S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. UNBLOCK LABS WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER’S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED “AS IS” AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT UNBLOCK LABS’S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST UNBLOCK LABS WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.
THE REPRESENTATIONS AND WARRANTIES OF UNBLOCK LABS CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST UNBLOCK LABS WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

Unblock Labs

An EatTheBlocks Company

unblock-labs.com