PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For BetSwirl (B2B)

13 December 2024

paladinsec.co

info@paladinsec.co

Table of Contents

Disclaimer

Paladin Blockchain Security ("Paladin") has conducted an independent audit to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocation for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies. Further, this audit report shall not be disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by Paladin.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided 'as is', and Paladin is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will Paladin or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

Cryptocurrencies and any technologies by extension directly or indirectly related to cryptocurrencies are highly volatile and speculative by nature. All reasonable due diligence and safeguards may yet be insufficient, and users should exercise considerable caution when participating in any shape or form in this nascent industry.

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the contracts are functioning as intended, specifically that the functions therein contained within said contracts have the desired intended effects, functionalities and outcomes of the Project team.

Paladin retains the right to re-use any and all knowledge and expertise gained during the audit process, including, but not limited to, vulnerabilities, bugs, or new attack vectors. Paladin is therefore allowed and expected to use this knowledge in subsequent audits and to inform any third party, who may or may not be our past or current clients, whose projects have similar vulnerabilities. Paladin is furthermore allowed to claim bug bounties from third-parties while doing so.

1 Overview

This report has been prepared for BetSwirl on the Avalanche, Arbitrum One, BNB Chain, Polygon and Base networks. Paladin provides a user-centred examination of the smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

1.1 Summary

Project Name	BetSwirl
URL	https://www.betswirl.com/
Platform	Avalanche, Arbitrum One, BNB Chain, Polygon and Base
Language	Solidity
Preliminary Contracts	https://github.com/BetSwirl/contracts-v2/commit/457f2a1bfb091d12a5c593835aeea485ab6ce30d
Resolution 1	https://github.com/BetSwirl/contracts-v2/commit/25e4600b9a5089a030bde9e141a096809facdb95

1.2 Contracts Assessed

Name	Contract	Live Code Match
FreeBet	0x7a1EFD33f41150E3247F14209b2a733bc6B1cb7a	MATCH
LeaderboardHub	0x0E5C8EA20a1EB26e5dDE5AFab5279F546dB92a79	MATCH

The contract addresses are on the same on all networks — we have only checked that the contracts on the Arbitrum network matches the audited code. Users should do the same check on the other networks.

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
Governance	0	-	-	-
High	0	-	-	-
Medium	2	2	-	-
Low	7	5	2	-
Informational	19	12	-	7
Total	**28**	**19**	**2**	**7**

Classification of Issues

Severity	Description
Governance	Issues under this category are where the governance or owners of the protocol have certain privileges that users need to be aware of, some of which can result in the loss of user funds if the governance's private keys are lost or if they turn malicious, for example.
High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
Medium	Bugs or issues that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
Informational	Consistency, syntax or style best practices. Generally pose a negligible level of risk, if any.

1.3.1 FreeBet

ID	Severity	Summary	Status
01	MEDIUM	Use of `safeIncreaseAllowance()` can brick the `FreeBet` contract for tokens requiring 0 approvals prior to approving a non-zero amount	✓ RESOLVED
02	LOW	Custom signature scheme is missing some important parameters such as contract address, which would be covered in EIP-712	PARTIAL
03	LOW	Unused bet amounts are not returned to the affiliate atomically	PARTIAL
04	INFO	Affiliates can DOS select players	ACKNOWLEDGED
05	INFO	Extra VRF fees paid in `wager()` are not returned to the user, but rather sent to the `FreeBet` contract	ACKNOWLEDGED
06	INFO	Insufficient validation	✓ RESOLVED
07	INFO	Accounting of balances will not work for rebasing tokens or tokens with a fee on transfer	✓ RESOLVED
08	INFO	`withdrawStuckFunds` can drain the balance if there is a double entrypoint token, or used on a chain like CELO where the native token can be treated as ERC20	ACKNOWLEDGED

1.3.2 LeaderboardHub

ID	Severity	Summary	Status
09	MEDIUM	`pullUnclaimedNFTs()` can result in affiliates stealing NFTs which are re-deposited in future leaderboards	RESOLVED
10	LOW	Lack of upper limit on `expirationTime`, as well as unsafe casting when calculating `expiresAt` can break intended logic	RESOLVED
11	LOW	Affiliates cannot unclaim any tokens if the leaderboard is not explicitly canceled or finalized	RESOLVED
12	LOW	A VRF call is not needed when only one winner is determined during finalization	RESOLVED
13	LOW	Incorrect validation for checking against the `randomNFTsLimit` in `createLeaderboard()`	RESOLVED
14	LOW	Using unchecked for calculating `totalShares` in `createLeaderboard()` is unsafe	RESOLVED
15	INFO	Contract charges a fixed fee while actual VRF costs vary based on the number of random NFTs	ACKNOWLEDGED
16	INFO	Unsafe casting without prior validation in multiple cases, such as with winners and playerIndex in `fulfillRandomWords()`	RESOLVED
17	INFO	Leaderboard manager cannot cancel leaderboard	RESOLVED
18	INFO	Users can have less time than expected to claim their random NFT rewards	ACKNOWLEDGED
19	INFO	Affiliate must wait for finalized leaderboard with no winners to expire in order to unclaim	RESOLVED
20	INFO	Gas optimizations	RESOLVED
21	INFO	Using `uint32` for timestamps without explicitly handling downcasting can result in issues when that year is reached	ACKNOWLEDGED
22	INFO	Duplicate winner addresses in the `winners_` array in `finalizeLeaderboard()` can result in overwriting `winnerPositions` mapping for a given winner	RESOLVED
23	INFO	Affiliate should not be `address(0)`	RESOLVED
24	INFO	Typographical issues	RESOLVED
25	INFO	Event indexing	RESOLVED
26	INFO	Users can create leaderboards that end in the same block	RESOLVED
27	INFO	`LeaderboardHub` functionality could break if VRF version becomes outdated	ACKNOWLEDGED
28	INFO	Fee-on-transfer and rebasing tokens not supported	RESOLVED

2 Findings

2.1 FreeBet

`FreeBet` implements logic for an affiliate to supply the tokens for a user to create bets, meaning they effectively get to place a free bet. However, the users are still required to pay for the bet's VRF fees in the native token. Generally, an affiliate will first deposit tokens into the contract which they can later designate for specific users to create bets. This designation comes from the manager signing data in a given `FreeBetData` struct, which includes the player who will get the free bet and the amount they are able to bet, among other information, allowing the user to get their free bet. The user can then call `wager` with this signature and specify any valid game.

There is also functionality to support withdrawing funds sent to this contract not from affiliate deposits. This includes any overpaid VRF fees which are sent back to the `FreeBet` contract rather than to the user who called `wager`.

In this system, the manager is a trusted address which has full control over all affiliate funds, with the exception of withdrawing them, meaning if the keys for this address were compromised, the affiliate funds would be at risk.

2.1.1 Privileged Functions

- `setManager[OWNER]`
- `setBank[OWNER]`
- `withdrawStuckFunds[OWNER]`

2.1.2 Issues & Recommendations

Issue #01	**Use of `safeIncreaseAllowance()` can brick the `FreeBet` contract for tokens requiring 0 approvals prior to approving a non-zero amount**
Severity	MEDIUM SEVERITY
Description	Some tokens do not allow approvals of a non-zero amount when the current approved amount is non-zero, such as USDT on Ethereum. This is an issue with the current flow when calling the `wager()` function, as this attempts to approve the exact amount which a user specifies for a game:

```
uint256 amount = freeBetData.amount;
[...]
if (!isGasToken) {
    IERC20(freeBetData.token).safeIncreaseAllowance(game,
amount);
}
```

Ultimately, a bet will be triggered which calls `Game:_newBet()` that can possibly decrease the `betAmount` based on `maxBetAmount` set for this game:

```
(
    bool isAllowedToken,
    uint256 maxBetAmount,
    uint256 maxBetCount
) = BANK.getBetRequirements(tokenAddress, multiplier);

[...]
if (tokenAmount > maxBetAmount) {
    if (isGasToken) {
        [...]
    }
    tokenAmount = maxBetAmount;
}

[...]

if (!isGasToken) {
    IERC20(tokenAddress).safeTransferFrom(
        msg.sender,
        address(this),
        tokenAmount * betCount
    );
}
```

	If `tokenAmount` is less than `maxBetAmount`, the actual amount transferred from the `FreeBet` contract will be less than what was approved in the original `wager()` call, leaving the approval at > 0. Then in the next `wager()` call, the non-zero approved amount for this token will cause the `safeIncreaseAllowance()` call to revert, bricking the functionality for these tokens.
Recommendation	Consider replacing `safeIncreaseAllowance()` on line 92 with `forceApprove()`.
Resolution	RESOLVED `forceApprove()` is now used. **Note:** `safeApprove(game, 0)` was introduced to ensure unused approval amounts are zeroed out, which we noted might have an edge case for tokens which do not allow approving 0. Team has acknowledged this.

Issue #02	**Custom signature scheme is missing some important parameters such as contract address, which would be covered in EIP-712**
Severity	LOW SEVERITY
Description	The `FreeBet` contract has implemented its own signature scheme for validating information about free bets, which fails to include some important information otherwise covered in the EIP-712 scheme, such as the contract address. This is relevant if, for example, the team decides to deploy another contract in the future which follows the same signature schema, it would be possible that signatures can be replayed on both contracts.
Recommendation	Consider replacing the custom signature scheme with EIP-712 signatures.
Resolution	PARTIALLY RESOLVED The current signature scheme is still used, but `freebetAddress` has been added to the `FreeBetData` struct for additional validation.

Issue #03	Unused bet amounts are not returned to the affiliate atomically
Severity	LOW SEVERITY

Description	Similar to Issue #01, it is possible that the amount a user requests to bet is higher than the `maxBetAmount`, which would result in less tokens being taken from the `FreeBet` contract than was decremented from the affiliate's balance:
	`unchecked {`
	`  affiliateBalances[freeBetData.affiliate][`
	`  freeBetData.token`
	`  ] -= amount;`
	`  totalAffiliateBalance[freeBetData.token] -= amount;`
	`}`
	
	This effectively means that the affiliate 'loses' these tokens, unless the admin later calls `withdrawStuckFunds()` and sends the funds back to the affiliate. Even if this occurs, we believe this is suboptimal as it results in wasted gas and potential delays.

Recommendation	We believe the refund should be handled within `wager()`, and can potentially be done by checking the change in token balance before and after the `wagerWithData()` call. Note: there is some complexity here when the betting token is `address(0)` and there are also VRF refunds sent back to the `FreeBet` contract.
Resolution	PARTIALLY RESOLVED This has been resolved for ERC20 tokens, but not for the gas token since it is not possible to know if the returned gas amount from the game contract is due to excess amount of VRF fees or due to game specific amount constraints without modifying game contracts that have already been deployed.

Issue #04	Affiliates can DOS select players
Severity	INFORMATIONAL
Description	Anybody can deposit funds to an affiliate that can be used by a player for a free bet. Affiliates however, can withdraw the whole balance that was assigned to them any time, which allows them to front-run select players that call `wager()` on chains with public mempools, withdraw their balance and as a result DOS the players.
Recommendation	Consider adding a delay withdrawal period in order to prevent affiliates from DOS-ing players.
Resolution	ACKNOWLEDGED This has been documented in the comments.

Issue #05	**Extra VRF fees paid in `wager()` are not returned to the user, but rather sent to the FreeBet contract**
Severity	INFORMATIONAL
Description	When a new bet for any PVH game is triggered, the extra VRF fees paid will be refunded to the caller, which in this case would be the FreeBet contract itself rather than the user who calls `wager()`. This is a known issue with the current implementation solving this by having a `withdrawStuckFunds()` function that allows the admin to remove these funds and potentially send them back to the original caller at a later time. We believe this is suboptimal, as these extra steps would waste gas and could possibly be implemented in the function call itself.
Recommendation	At a minimum, document this in the natspec for the `wager()` function. This could also potentially be solved by determining whether any excess native tokens were sent to the `FreeBet` contract after the call to `wagerWithData()`, which would indicate a refund was provided. Note: There is some complexity when the token used is `address(0)` as it is possible the entire bet was not used as well.
Resolution	ACKNOWLEDGED This has been documented in the comments.

Issue #06	Insufficient validation
Severity	INFORMATIONAL
Description	There is no validation with `deposit()` that the amount provided is non-zero. Consider adding this check, and reverting if the amount is 0. — Within `wager()`, there is no validation that the token used is valid. This validation occurs much later in the `_newBet()` call, meaning a lot of wasted gas just to find out your bet is invalid. Consider checking `Bank:getBetRequirements()` earlier on.
Recommendation	Consider implementing the above mentioned recommendations
Resolution	RESOLVED Validation for `amount` is added to the codebase.

Issue #07	Accounting of balances will not work for rebasing tokens or tokens with a fee on transfer
Severity	INFORMATIONAL
Description	The method of accounting for token balances assumes that balances in the contract will never change once deposited. This assumption breaks for rebasing tokens or tokens with a fee on transfer.
Recommendation	Consider making this explicitly clear in the natspec if the intention is to never support such tokens. Otherwise, the accounting will need to be re-designed.
Resolution	RESOLVED Comment added in the codebase stating these tokens are not supported.

Issue #08	**`withdrawStuckFunds` can drain the balance if there is a double entrypoint token, or used on a chain like CELO where the native token can be treated as ERC20**
Severity	INFORMATIONAL
Location	`withdrawStuckFunds(celoERC20, totalBalance, to)`
Description	`withdrawStuckFunds` can drain the balance if there is a double entrypoint token on the chains where the protocol is deployed. On chains like CELO where the native token can be treated as an ERC20 token, and in the case where only native token deposits are used, `totalAffiliateBalance[celoERC20]` would be 0, but `totalBalance` would be the balance of native CELO tokens. Ref: https://github.com/Uniswap/v4-core/pull/779 We understand that the team will not deploy on CELO at the time of this audit, but we have included this issue for their awareness.
Recommendation	This issue is mainly added for informational purposes as we do not see an ideal resolution except to not support such chains or tokens.
Resolution	ACKNOWLEDGED

2.2 LeaderboardHub

`LeaderboardHub` implements logic for users / affiliates to create leaderboards, which effectively pay out tokens and NFTs to winners which are set by the `LEADERBOARD_MANAGER`. There are both static NFTs and random NFTs, where the static NFTs are given to winners based on their position, while the random NFTs are sent to winners at random. This randomness is implemented using Chainlink VRF, where the `LEADERBOARD_MANAGER` is required to pay the VRF fees.

When a leaderboard is created, it has a fixed `endsAt` timestamp where it can be finalized. After it is finalized, there is an `expirationTime` duration during which winners are able to withdraw their winnings (NFTs and tokens) prior to the affiliate being able to withdraw tokens or NFTs that were not claimed.

In this flow, the address with the `LEADERBOARD_MANAGER` role has full control over determining the winners for a leaderboard, in turn meaning they have full control over the assets in any given leaderboard.

2.2.1 Privileged Functions

- `finalizeLeaderboard[LEADERBOARD_MANAGER]`
- `cancelLeaderboard[OWNER]`
- `withdrawFees[OWNER]`
- `setFee[OWNER]`
- `setRandomNFTsLimit[OWNER]`
- `setMinExpirationTime[OWNER]`
- `setChainlinkConfig[OWNER]`

2.2.1 Issues & Recommendations

Issue #09	**`pullUnclaimedNFTs()` can result in affiliates stealing NFTs which are re-deposited in future leaderboards**
Severity	MEDIUM SEVERITY

Description	The pullUnclaimedNFTs() function handles removing both unclaimed static NFTs and random NFTs. However, there is an issue with the logic for handling random NFTs, where the `nftAvailable` mapping for these NFTs is incorrectly updated:
	`if (`
	`  nftAvailable[leaderboardId][position][collection][id] ||`
	`  position == 0`
	`) _unclaimNFT(leaderboardId, 0, collection, id, to);`
	
	For the random NFTs, the `_unclaimNFT()` function is passed 0 as the position argument, meaning the incorrect position is being deleted:
	`delete nftAvailable[leaderboardId][position][collection][id];`
	
	This in turn means that the actual position for this NFT is still set to true in the `nftAvailable` mapping. Now let's assume this NFT was removed and sold to another user, who now adds it into a new leaderboard. The affiliate of the original leaderboard can then call `pullUnclaimedNFT()` to steal this NFT from the other leaderboard. This is possible because the following check will pass, due to the mapping not being correctly updated:
	`if (!nftAvailable[leaderboardId][position][collection][id])`
	`  revert NothingToClaim();`

Recommendation	Within `pullUnclaimedNFTs()`, make sure to pass in the correct position to delete when the NFT's position is not 0.
Resolution	RESOLVED `position` is now being passed instead of 0.

Issue #10	**Lack of upper limit on `expirationTime`, as well as unsafe casting when calculating `expiresAt` can break intended logic**
Severity	LOW SEVERITY
Description	When a leaderboard is created, the provided `expirationTime` must be above `minExpirationTime` but there is no such constraint that prevents the user from providing an extremely large expiration time which in practice can result in the affiliate being unable to collect unclaimed assets. Additionally, there is another potential issue due to overflow when `createLeaderboard()` is called, which can result in leaderboards being immediately expired. The overflow issue lies in this line: `leaderboards[leaderboardId].expiresAt = uint32(block.timestamp) + leaderboard.expirationTime;` `leaderboards[leaderboardId].expiresAt` is indeed `uint40` but when we combine two `uint32` values, their result is stored in `uint32` and then assigned to `uint40` (in our case). When there is an overflow due to adding these two values, it can result in `expiresAt` immediately being less than `block.timestamp`.
Recommendation	Implement maximum expiration time constraint and make sure that overflows will not occur.
Resolution	RESOLVED An upper limit on `expirationTime` has been added in `createLeaderboard()` and `block.timestamp` is now first cast to `uint40`.

Issue #11	**Affiliates cannot unclaim any tokens if the leaderboard is not explicitly canceled or finalized**
Severity	LOW SEVERITY
Description	While anyone can create a leaderboard and transfer tokens/NFTs as rewards, these assets cannot be retrieved without input from a manager or admin. If the leaderboard remains unfinalized or is not properly canceled, the funds will remain stuck in the contract.
Recommendation	Consider implementing a time-based recovery mechanism that allows affiliates to unclaim the deposited assets as reward if the leaderboard remains inactive. This recovery option would activate after a specified period without interaction (finalization or cancellation).
Resolution	RESOLVED Logic has been added which allows this.

Issue #12	**A VRF call is not needed when only one winner is determined during finalization**
Severity	LOW SEVERITY
Description	Within `finalizeLeaderboard()`, a random word from Chainlink VRF is requested if `randomNFTsCount` and `winnersCount` are both more than 1. However, when a single winner is determined, all random NFTs should be assigned to this winner, and the random functionality can be bypassed in this particular case. The logic for random NFTs assignment to specific positions and their availability can be executed within `finalizeLeaderboard()` on the spot.
Recommendation	Consider assigning the random NFTs to the winner position within `finalizeLeaderboard()` when only one winner is determined, so that VRF related expenses from the configured Chainlink `subId` account can be saved.
Resolution	RESOLVED There is now a separate flow for handling the case where there is a single winner.

Issue #13	Incorrect validation for checking against the `randomNFTsLimit` in `createLeaderboard()`
Severity	LOW SEVERITY
Description	Within `createLeaderboard()`, the number of unique collections for random NFTs is being checked against the `randomNFTsLimit` instead of the actual number of NFTs. This issue is significant because these NFTs must be paid for by the `LEADERBOARD_MANAGER` (VRF costs). If they decline payment, the admin would have to call `cancelLeaderboard()`, after which the affiliate would need to withdraw all assets and attempt the process again. A considerable amount of gas is wasted due to this incorrect validation.
Recommendation	Consider counting the actual number of random NFTs when looping over the user input, and compare this final sum against `randomNFTsLimit`.
Resolution	RESOLVED The validation method has been updated.

Issue #14	Using unchecked for calculating `totalShares` in `createLeaderboard()` is unsafe
Severity	LOW SEVERITY
Description	It is possible for the sum of the `_shares` array to overflow, resulting in exceedingly high values for the shares, while being a low value for `totalShares`. This will result in users being unable to claim token rewards.
Recommendation	Consider removing the `unchecked` block.
Resolution	RESOLVED

Issue #15	Contract charges a fixed fee while actual VRF costs vary based on the number of random NFTs
Severity	INFORMATIONAL
Description	Within the `finalizeLeaderboard` function, the `callbackGasLimit` is increased with `callbackGasExtraNFT` for each random NFT. The bigger the `callbackGasLimit`, the more the configured Chainlink account will get charged. The contract's fixed fee structure for leaderboard creation may result in incorrect charging. Users can be overcharged or undercharged based on the number of random NFTs that are provided and the buffer that is configured in the admin's fixed fee.
Recommendation	Consider increasing the configured fee based on the amount of random NFTs to be able to charge users more accurately.
Resolution	ACKNOWLEDGED The team stated that this fee is not specifically related to only VRF costs. They will be actively managing this variable to adjust fees.

Issue #16	**Unsafe casting without prior validation in multiple cases, such as with winners and `playerIndex` in `fulfillRandomWords()`**
Severity	INFORMATIONAL
Description	Unsafe casting operations have been identified where variables are downcast without prior validation of size compatibility. For instance, `playerIndex`, which can reach the length of the winners array, is downcast to `uint16`. While it is unlikely that the winners array would exceed 2^16-1 entries, no explicit validation is being performed. In a worst-case scenario, winners positioned in the latter portions of the winners array may be prevented from receiving random NFTs due to the `playerIndex` downcasting. Although this scenario is highly improbable due to gas constraints, future-proofing is preferred over relying on current limitations.
Recommendation	For all variables that are downcast, consider adding explicit checks that determine the downcast is safe. For the winners array, its length should be checked in the `fulfillRandomWords()` function call.
Resolution	RESOLVED Within `createLeaderboard()`, the length of `shares_` is checked to not be larger than `type(uint16).max - 1`.

Issue #17	**Leaderboard manager cannot cancel leaderboard**
Severity	INFORMATIONAL
Description	The leaderboard manager role can only call `finalizeLeaderboard()`. Leaderboard managers should be able to call `cancelLeaderboard()` as this is a leaderboard-related operation while the default admin role should be responsible only for the global contract variables.
Recommendation	Consider replacing the `onlyRole(DEFAULT_ADMIN_ROLE)` with `onlyRole(LEADERBOARD_MANAGER)` in `cancelLeaderboard()`.
Resolution	RESOLVED `LEADERBOARD_MANAGER` is now validated for this function.

Issue #18	Users can have less time than expected to claim their random NFT rewards
Severity	INFORMATIONAL
Description	The expiration period is initiated after leaderboard finalization. Winners' ability to claim random NFTs is dependent on the `fulfillRandomWords()` VRF call, which can be completed within 0-200 confirmed blocks (representing minutes to an hour on most chains). When Chainlink is configured to operate with Link tokens instead of native ones, the Link price feed may become stale, causing the `fulfillRandomWords()` call to be reverted. This situation can reduce the effective claiming period for winners, as the expiration countdown is started at finalization rather than when rewards become claimable.
Recommendation	Consider recording when the leaderboard was finalized, and increase the `expiresAt` property inside `fulfillRandomWords()`.
Resolution	ACKNOWLEDGED

Issue #19	**Affiliate must wait for finalized leaderboard with no winners to expire in order to unclaim**
Severity	INFORMATIONAL
Description	According to this code comment above `finalizeLeaderboard()`: "`winners_` length could be lower than `shares` length in the case where not enough players are eligible." The winners can be less if there are not enough eligible players. If there is no winner, the `finalizeLeaderboard` function will set `expiresAt` to `now + expirationTime` but this is not needed since there are no winners that need time to claim rewards.
Recommendation	Consider setting `leaderboards[leaderboardId].expiresAt` to `block.timestamp - 1` when there are no winners during leaderboard finalization in order to allow affiliates to unclaim straight away.
Resolution	RESOLVED

Issue #20	Gas optimizations
Severity	INFORMATIONAL
Description	Line 386 Consider reverting when the position is 0 in `claimNFTs()` to save the gas cost of looping through every NFT. — Consider placing position and collection together in the `NFTReward` struct for better packing. — Iterators throughout the codebase use non-`uint256` data types which results in extra gas costs, as well as being more unsafe in general (line 195, line 198, line 355, etc). Consider converting all iterator data types to `uint256`. Generally, gas is only saved when using smaller data types when they are placed in storage. — Consider placing `_leaderboardsCount`, `randomNFTsLimit`, and `minExpirationTime` together for better packing in the `LeaderboardHub` contract. — Consider placing `nativePayment` at the top of the `ChainlinkConfigSet` struct for better packing. — `nftCollection` on line 200 can be cached, as it will be the same for all the ids being looped over.
Recommendation	Consider implementing the above mentioned recommendations
Resolution	RESOLVED

Issue #21	**Using `uint32` for timestamps without explicitly handling downcasting can result in issues when that year is reached**
Severity	INFORMATIONAL
Description	The logic for this contract will not work properly once the block timestamp overflows the `uint32` data type.
Recommendation	Consider using at least `uint48` for casting timestamps.
Resolution	ACKNOWLEDGED

Issue #22	**Duplicate winner addresses in the `winners_` array in `finalizeLeaderboard()` can result in overwriting `winnerPositions` mapping for a given winner**
Severity	INFORMATIONAL
Description	This is a known issue, with there being no extra validation added to check for duplicates at this time. However, there is no documentation for this in the codebase at the moment.
Recommendation	At a minimum, document this in the natspec for the `finalizeLeaderboard()` function.
Resolution	RESOLVED Comment has been added to the codebase.

Issue #23	Affiliate should not be `address(0)`
Severity	INFORMATIONAL
Description	There should be a check within `createLeaderboard()` to ensure the provided affiliate is not `address(0)`, otherwise funds cannot be unclaimed since the caller must be the affiliate.
Recommendation	Consider implementing the check.
Resolution	RESOLVED

Issue #24	Typographical issues
Severity	INFORMATIONAL
Description	In withdrawFees() an event is emitted even if funds are not withdrawn from the contract. Consider emitting events only if feesToWithdraw != 0. — IVRFCoordinatorV2Plus import in LeaderboardHub.sol can be removed as it is not used anywhere.
Recommendation	Consider resolving the typographical issues.
Resolution	RESOLVED

Issue #25	Event indexing
Severity	INFORMATIONAL

Location	`event Claimed(`
	`  address player, // should be address indexed player for this example`
	`  uint256 leaderboardId,`
	`  address token,`
	`  uint256 amount`
	`);`

Description	Ensure all events within `ILeaderboardHub` use the `indexed` keyword where it is needed in order to facilitate easy off-chain querying of specific data. For example, when the player parameter in Claimed events is not indexed, frontends or backends must retrieve all Claimed events ever emitted and filter them locally to find events associated with a specific player.
Recommendation	Consider implementing the indexing of the event arguments where needed.
Resolution	RESOLVED

Issue #26	Users can create leaderboards that end in the same block
Severity	INFORMATIONAL
Description	The documentation points out that bets after `endsAt` should not be counted. The check in `createLeaderboard()` ensures that endsAt is not in the past but allows for `block.timestamp` to be endsAt: `if (block.timestamp > endsAt)`
Recommendation	Consider implementing a minimum amount of time that needs to pass to consider the leaderboard as ended.
Resolution	RESOLVED Validation was added so the function reverts if `endsAt` is equal to `block.timestamp`.

Issue #27	**`LeaderboardHub` functionality could break if VRF version becomes outdated**
Severity	INFORMATIONAL
Description	According to Chainlink documentation (https://docs.chain.link/vrf/release-notes#2024-07-15-vrf-v2-and-v1-deprecation-announcement), VRF v1 and v2 will become deprecated on 29.11.2024. This could be the case for VRF v2.5 in the future if future versions of VRF come out. For this reason, immutable contracts like `LeaderboardHub` should take this possibility into account.
Recommendation	Immutable contracts should be insulated from directly interacting with Chainlink VRF. One way to achieve this is to create a separate `VRFHandler` contract that acts as a bridge between immutable contracts and Chainlink VRF. Another solution is making `LeaderboardHub` upgradeable.
Resolution	ACKNOWLEDGED

Issue #28	**Fee-on-transfer and rebasing tokens not supported**
Severity	INFORMATIONAL
Description	The logic does not support the use of fee-on-transfer and rebasing tokens.
Recommendation	If this is expected, consider at least documenting this within the natspec. Otherwise, the logic will need to be updated.
Resolution	RESOLVED A comment has been added.

PALADIN
BLOCKCHAIN SECURITY