SB SECURITY

Mizuchi

Security Review

Jan 10, 2025

Conducted by:
Blckhv, Lead Security Researcher
Slavcheww, Lead Security Researcher

Contents

1. About SBSecurity

SBSecurity is a duo of skilled smart contract security researchers. Based on the audits conducted and numerous vulnerabilities reported, we strive to provide the absolute best security service and client satisfaction. While it's understood that 100% security and bug-free code cannot be guaranteed by anyone, we are committed to giving our utmost to provide the best possible outcome for you and your product.

Book a Security Review with us at sbsecurity.net or reach out on Twitter @Slavcheww.

2. Disclaimer

A smart contract security review can only show the presence of vulnerabilities **but not their absence**. Audits are a time, resource, and expertise-bound effort where skilled technicians evaluate the codebase and their dependencies using various techniques to find as many flaws as possible and suggest security-related improvements. We as a company stand behind our brand and the level of service that is provided but also recommend subsequent security reviews, on-chain monitoring, and high whitehat incentivization.

3. Risk classification

	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

3.1. Impact

- **High** - leads to a significant loss of assets in the protocol or significantly harms a group of users.
- **Medium** - leads to a moderate loss of assets in the protocol or some disruption of the protocol's functionality.
- **Low** - funds are not at risk.

3.2. Likelihood

- **High** - almost **certain** to happen, easy to perform, or highly incentivized.
- **Medium** - only **conditionally possible**, but still relatively likely.
- **Low** - requires specific state or **little-to-no incentive**.

3.3. Action required for severity levels

- High - **Must** fix (before deployment if not already deployed).
- Medium - **Should** fix.
- Low - **Could** fix.

4. Executive Summary

Overview

Project	Mizuchi
Repository	Private
Commit Hash	c6823f25b562793405431d194009cd53e3362fc0
Resolution	ef53dbd97bd96cab3f9e3a6aecb969815d53b056
Timeline	Audit: December 21 - December 27, 2024 Mitigation: December 27, 2024 - January 10, 2025

Scope

Mizuchi.sol
MizuchiAuctions.sol
MizuchiBuyAndBurn.sol
MizuchiHoard.sol
MizuchiInjector.sol
MizuchiNFT.sol

Issues Found

Critical Risk	0
High Risk	2
Medium Risk	8
Low/Info Risk	30

5. Findings

5.1. High severity

5.1.1. Strict slippage check will DoS the deployLp

Severity: High Risk

Description: `MizuchiBuyBurn::mintInitialPosition` always expects all the X28 and Mizuchi tokens to be supplied to the LP, but due to the roundings that happen in UniswapV3's math, there are scenarios where at least 1 wei of either of the tokens can remain in the BnB contract.

In that scenario, deployment will be reverting, and trading won't be able to be enabled due to the strict slippage expecting the entire `initialLiquidityAmount` to be supplied.

For this issue to occur, the only thing we need is the accumulated `X28` to be less than the `LP_POOL_SIZE.`

Recommendation: Change the `amount0Min` and `amount1Min` to be 80/90% of the `initialLiquidityAmount` or pass them as parameters.

Resolution: Fixed

5.1.2. Anyone can set the initial sqrtPriceX96 of Mzi/X28 pool and grief the protocol

Severity: High Risk

Description: Anyone can deploy the X28/Mzi pair beforehand and set any arbitrary `sqrtPriceX96` he wants, which will cause the following issues to happen:

1. trading won't be enabled due to the strict slippage, unlike [H-01], here, more than 1 wei will be left, in fact, after the tests we performed, the entire X28 remains locked in the `MizuchiBuyBurn` contract.
2. price will be skewed and will deviate from the desired 1:1 for Mzi/X28.

An attacker can achieve this with 2 attacks:

1. He can perform a swap, even when the Mzi token is non-transferable - this will move the ticks to either lower or higher end, depending on the token orders in the pool. If `X28` is token1, he will move the ticks to the higher end. This will change the `sqrtPriceX96`.
2. Same swap as 1. + single side supply of `X28`, this will raise the price even more.

Point 2. is possible since UniV3 has concentrated liquidity, and if the attacker moves the ticks to the higher end, then any tick range below the actual tick will require only `X28` for the LP (assuming X28 is token1).

Recommendation: Pool should be initialized and funded in the Mizuchi constructor or custom fixPool function should be added.

1. First check if the pool has been created beforehand.
2. Check if there are tokens supplied and tick is different from the intended (0). For tokens supplied: liquidity variable of the Pool > 0, this checks whether there is X28 supplied. For tick: tick ≠ 0, this changed means there is X28 supplied
3. You will have to do the opposite swap. The caveat of these empty swaps is that no tokens will be pulled from the caller, only `sqrtPriceX96` will be changed.

Resolution: Partially fixed - The team has decided to add a require statement, which will revert in case the price has moved from the desired. If that happens, they will transfer the ownership of MZI and bundle a reversed swap with the deployLP to prevent frontruns.

5.2. Medium severity

5.2.1. Approval not cleared after UniswapV3 operations

Severity: Medium Risk

Description: `_swapX28ToMizuchi` and `_swapEthToTitanX` both perform exact output swaps. However they only refund the caller without clearing the leftover approvals of tokenOut. Furthermore `safeIncreaseAllowance` is used which will increase the allowance instead of setting it to certain number each time. These 2 in combination will result in ever increasing allowance.

Another instance is the `MizuchiInjector::_addToLp` where it's not guaranteed that all the tokens will be taken, also doesn't reset the approval to the `NFPM`.

Recommendation: Make sure to clear the approvals after swap and liquidity increase happens.

Resolution: Acknowledged

5.2.2. Mzi/TitanX fees completely lost

Severity: Medium Risk

Description: It was stated in the docs that fees from the Mzi/TitanX position would be compounded forever, but there is no such logic in `MizuchiInjector`. Currently, there is a wrong assumption that fees accumulated will be used as liquidity automatically. However, this is not the case in UniV3, and fees must be manually reinvested by collecting them and then supplying them to the pool.

Recommendation: Add the missing logic in `_addToLp` to collect the fees and include them in the `increaseLiquidity` params. Also, a view function the check the fees owed should be introduced so it's easier for the callers to estimate the amounts min required.

Resolution: Fixed

5.2.3. Anyone can make huge deposits and cause smaller users not receive auction rewards

Severity: Medium Risk

Description: Calculation for the auction day payout in `_calculateAuctionPayout` is done the following way:

```
//user's deposit * MZI allocation (b/n 3% and 20% of the Hoard's balance) / total X28 deposited for the day
depositAmount * stats.allocation) / stats.totalDeposited
```

As we can see, when there is a huge auction participation, i.e., a lot of `X28` deposited, there is a possibility that a slighter player will not receive any payout due to the denominator being bigger than the nominator, which will result in rounding to 0.

Since there are no constraints on the minimum and maximum entry amount, any number b/n one wei and uint256.max can be deposited, so the chance for some participators to lose their payouts is relatively big.

Recommendation: Consider increasing the precision of the deposit amount at the time of entering the auction and decreasing it by the same amount when claiming the payout.

Resolution: Acknowledged

5.2.4. _addNftsToWyrmhole will do another external call for NFT transfer before day 34

Severity: Medium Risk

Description: Transfer of the MizuchiNFT from the caller to the escrow before day 34 will always perform another external call since `safeTransferFrom` is used with `this.` prefix. Using that adds a side effect that `msg.sender` is changed from the caller (NFT owner) to the MizuchiNFT itself, and approval will be needed, while if this was not used, NFTs will be able to be transferred from the owner to escrow without giving explicit approval.

Recommendation: In order to simplify and reduce the gas costs, remove the `this.` prefix but only from the `_addNftsToWyrmhole`. `_removeNftsFromWyrmhole` should remain unchanged. Otherwise, self-approval from the `MizuchiNFT` will be needed.

Resolution: Fixed

5.2.5. `Mizuchi::distribute` is missing incentive

Severity: Medium Risk

Description: Excess X28 after deploying the LPs will accumulate in the `Mizuchi` and should be manually distributed from the `Mizuchi::distribute` function. The problem is that there is no incentive given to the caller, and most likely, the owners themselves will be forced to call the function. Otherwise, no one will do it if they have to pay the gas without being rewarded.

Recommendation: Add incentive to the `Mizuchi::distribute`.

Resolution: Acknowledged

5.2.6. LP injection can be sandwiched due to slippage args being controlled by the caller

Severity: Medium Risk

Description: `MizuchiInjector::inject` exposes a user-controlled `amountDesiredMzi` and `amountDesiredTitanX`, unlike the swaps, here `twap` checks are missing, and users can sandwich any LP injection. However, there are no serious issues that will lead to loss of funds because the position will be in the full range of liquidity, and the attack where the tick is changed before the transaction in the other direction so these funds will not accrue any fees is not possible. One case that can lead to insufficient LPing is when `inject` can be frontran by performing a swap and changing the token ratio, the only harm will be that not all the tokens will be supplied, and part of them will be burned through `MizuchiInjector::burn.`

Recommendation: Consider adding TWAP to calculate the minimum amounts passed to the `IncreaseLiquidityParams` in `MizuchiInjector::_addToLp.`

Resolution: Acknowledged

5.2.7. Collected fees, waiting for injection can be burned from anyone

Severity: Medium Risk

Description: Anyone can collect and burn the collected fees from the UniswapV3 position in `MizuchiInjector` instead of re-compounding them. This will lead to decreased liquidity in the `TitanX/Mzi` pool and will deviate from the intended design outlined in the whitepaper, which says all the fees collected should be reinvested.

Recommendation: Remove the whole burning functionality. This will remove the possibility of malicious users to prevent injecting them.

Resolution: Fixed

5.2.8. Payout Pool starts at day 8 instead of 16

Severity: Medium Risk

Description: According to the documentation, the first payout cycle has to start on day 16 after the mint starts. However, it currently starts on the 8th day. That discrepancy will greatly deflate the amount of Mzi that will be distributed during the first payout cycle.

Recommendation: In the constructor, set the lastCycleTs to `MINT_END`

Resolution: Fixed

5.3. Low/Info severity

5.3.1. Excess X28/Mzi liquidity not refunded in initial position

Severity: Low Risk

Description: As explained in H-01, there are scenarios when not all the tokens on the initial liquidity will be taken. In that case, tokens will be locked in the contract. However, this doesn't pose a significant risk since both `X28` and `MZI` will be used in the BnB process. Yet another issue is that approvals to `NFPM` won't be cleared.

Recommendation: Make sure to clear the approvals after minting the initial position. If you don't want these tokens to be included in the BnB, transfer them back to the owner.

Resolution: Acknowledged

5.3.2. Lows, Informational issues and code suggestions

Description:

1. Parent variable shadowing: MizuchiHoard: owner, MizuchiNFT: _baseURI, MizuchiAuction: owner

2. pool check `_createPool` is unnecessary - manager already does the check so `X28_MIZUCHI_LP` directly can be assigned to the return value of `createAndInitializePoolIfNecessary`

3. In MizuchiNFT::deposit's natspec - Depoists Deposits

4. `MizuchiNFT::_processNftTiers` - currentTime can be moved to `_setNftMintData` directly

5. `MizuchiNFT`: add checks for account != address(0) in `getPackedTokenData` and `batchGetPackedTokenData`

6. `MizuchiHoard`: declare MIZUCHI as an IERC20, this will remove all the castings done below

7. `MizuchiBuyBurn`: remove `onlyToken` modifier from `_createPool`

8. cast `100` to `uint16(100)` for `increaseObservationCardinalityNext` in `BnB::_createPool`

9. `MizuchiBuyAndBurn::calculateCurrentInterval` minuscule X28 will be left due to the divisions performed (will be utilized in the next intervals) - [ACKNOWLEDGED]

10. in `MizuchiInjector:inject` increase `capPerCall` since it's only 59 cents now - [ACKNOWLEDGED]

11. in `MizuchiInjector` - `LEGENDX` not used

12. `MizuchiInjector::enableInjector` - use normal `transferFrom`

13. Replace all the safe ERC721 operations with non-safe, all the receivers will be aware they will be receiving tokens, since passing receiver is not possible, always `msg.sender` is the recipient

14. `MizuchiAuction::_getCurrentDay` will be reverting with underflow until mint doesn't end, no issue but bad UX

15. `MizuchiHoard` - not all the MZI will be allocated, daily allocation is capped at 20% of the balance max, if no new tokens enter the pool, it can be changed to give all the remaining balance and will drip small amounts - [ACKNOWLEDGED]

16. `MizuchiNFT` - use normal `mint` instead of `safeMint` for the NFT, callers know they will be receiving the NFTs

17. `MizuchiNFT::_calculateSerpentDue` - merge serpent paid and 26-day onward check

18. Early `Mzi/X28` BnB swaps can be manipulated - [ACKNOWLEDGED]

19. `MizuchiInjector::_addToLp` and `MizuchiBuyAndBurn::_mintInitialPosition` Approvals not removed after injecting LPs - [ACKNOWLEDGED]

20. Not used, commented 2nd `depositOf` mapping from `MizuchiAuctions` can be removed

21. `incentiveFeeBPS` in `MizuchiInjector` can be increased because now it does 2 swaps and 1 liquidity increase, after another problem fix it will also collect fees and all that for `0.3%`, users will not have the incentive to call it - [ACKNOWLEDGED]

22. Change `@param lpTokenId Token ID of the Uniswap V3 pair NFT.` to `@param lpTokenId Token ID of the Uniswap V3 TitanX/MZI pair NFT`, in `enableInjector()`'s natspec

23. In `MizuchiInjector:_addToLp()`, `token0` and `token1` are not used

24. Rename `x28Price` to `titanXPrice` in `mintWithTitanX` and `mintWithEth` - [ACKNOWLEDGED]

25. Use `BPS_BASE` instead of 10,000 in `MizuchiNFT._twapCheckExactOutput`

26. Use predefined mapping instead `_getTierPrice` - [ACKNOWLEDGED]

27. In `MizuchiBuyBurn::deployLP`, assign which token is token0 at the time of minting the position and rework the `_getPoolConfig` to not perform comparison every time but use the variable

28. First bnb call will enter the else case in calculateCurrentInterval and will loop for 2 days where the 1st day values will be 0s. Add case for the first call if necessary - [ACKNOWLEDGED]

29. Anyone can grief the payout pool by updating the cycle before calling BnB - [ACKNOWLEDGED]

30. Order of mint data packing is reversed, it should be: `tier, ts, multiplier`, but in `_packMintData` it's: `tier, multiplier, ts`.

Resolution: Fixed - 1,3,4,5,6,7,8,11,12,13,14,16,17,20,22,23,25,27,30