GUARDIAN AUDITS

SMART CONTRACT SECURITY AUDIT OF

MIMSwap

Summary

Audit Firm Guardian

Prepared By Owen Thurm, Daniel Gelfand, giraffe0x, Arnie Jimenez, 0xScourgedev, 0xnirlin

Client Firm MIMSwap

Final Report Date March 15, 2024

Audit Summary

MIMSwap engaged Guardian to review the security of its PMM exchange. From the 29th of February to the 11th of March, a team of 6 auditors reviewed the source code in scope. All findings have been recorded in the following report.

Notice that the examined smart contracts are not resistant to internal exploit. For a detailed understanding of risk severity, source code vulnerability, and potential attack vectors, refer to the complete audit report below.

🔗 Blockchain network: **Blast**

✅ Verify the authenticity of this report on Guardian's GitHub: https://github.com/guardianaudits

📊 Code coverage & PoC test suite: https://github.com/GuardianAudits/MimSwap-PoCs

Table of Contents

Project Information

Smart Contract Risk Assessment

Addendum

Project Overview

Project Summary

Project Name	MIMSwap
Language	Solidity
Codebase	https://github.com/Abracadabra-money/abracadabra-money-contracts
Commit(s)	Initial: ab3ab131c008422768312188d43e95a53191b241 Final: 5c16e196d2b17a458691c8ec730b9ae7babb0c68

Audit Summary

Delivery Date	March 15, 2024
Audit Methodology	Static Analysis, Manual Review, Test Suite, Contract Fuzzing

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
● Critical	1	0	0	0	0	1
● High	5	0	0	0	1	4
● Medium	7	0	0	2	0	5
● Low	15	0	0	3	0	12

Audit Scope & Methodology

Vulnerability Classifications

Severity	Impact: *High*	Impact: *Medium*	Impact: *Low*
Likelihood: *High*	🔴 Critical	🟠 High	🟡 Medium
Likelihood: *Medium*	🟠 High	🟡 Medium	🟢 Low
Likelihood: *Low*	🟡 Medium	🟢 Low	🟢 Low

Impact

High Significant loss of assets in the protocol, significant harm to a group of users, or a core functionality of the protocol is disrupted.

Medium A small amount of funds can be lost or ancillary functionality of the protocol is affected. The user or protocol may experience reduced or delayed receipt of intended funds.

Low Can lead to any unexpected behavior with some of the protocol's functionalities that is notable but does not meet the criteria for a higher severity.

Likelihood

High The attack is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount gained or the disruption to the protocol.

Medium An attack vector that is only possible in uncommon cases or requires a large amount of capital to exercise relative to the amount gained or the disruption to the protocol.

Low Unlikely to ever occur in production.

Audit Scope & Methodology

Methodology

Guardian is the ultimate standard for Smart Contract security. An engagement with Guardian entails the following:

- Two competing teams of Guardian security researchers performing an independent review.
- A dedicated fuzzing engineer to construct a comprehensive stateful fuzzing suite for the project.
- An engagement lead security researcher coordinating the 2 teams, performing their own analysis, relaying findings to the client, and orchestrating the testing/verification efforts.

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. Comprehensive written tests as a part of a code coverage testing suite.
- Contract fuzzing for increased attack resilience.

Invariants Assessed

During Guardian's review of MIMSwap, fuzz-testing with Echidna was performed on the protocol's main functions. Given the dynamic interactions and the potential for unforeseen edge cases in the protocol, fuzz-testing was imperative to verify the integrity of several system invariants.

During the engagement, the Echidna fuzzing suite, developed by 0xScourgedev, was employed for 65,000,000+ runs to assess specific invariants. The fuzzing suite targets the Router.sol, MagicLP.sol and Factory.sol contracts. PrivateRouter.sol and the functionality that is included with it was not fuzzed as part of this engagement as it was added during remediations. Additionally, the blast contracts were not fuzzed due to the timeboxed nature of this engagement.

ID	Description	Tested	Passed	Remediation	Run Count
GENERAL-01	Does not silent revert	✅	✅	✅	65M+
LIQ-01	If the base and quote token balance is 0, the amount of base tokens and quote tokens in the pool is always strictly increasing after adding liquidity	✅	✅	✅	65M+
LIQ-02	If the base and quote token balance is 0, the amount of base and quote tokens of the user is always strictly decreasing after adding liquidity	✅	✅	✅	65M+
LIQ-03	The total supply of lp tokens is always strictly increasing after adding liquidity	✅	✅	✅	65M+
LIQ-04	The lp token balance of the user is always strictly increasing after adding liquidity	✅	✅	✅	65M+
LIQ-05	The amount of base tokens and quote tokens in the pool is always decreasing after removing liquidity	✅	✅	✅	65M+
LIQ-06	The amount of base and quote tokens of the user is always increasing after removing liquidity	✅	✅	✅	65M+

Invariants Assessed

ID	Description	Tested	Passed	Remediation	Run Count
LIQ-07	The total supply of lp tokens is always strictly decreasing after removing liquidity	✅	✅	✅	65M+
LIQ-08	The lp token balance of the user is always strictly decreasing after removing liquidity	✅	✅	✅	65M+
LIQ-09	Base and quote tokens are never transfered to the user for free when removing liquidity	✅	✅	✅	65M+
LIQ-10	previewAddLiquidity() never reverts for reasonable values	✅	✅	✅	65M+
LIQ-11	previewRemoveLiquidity() never reverts for reasonable values if the total supply of lp tokens is greater than 0	✅	✅	✅	65M+
LIQ-12	Adding liquidity must provide less or equal shares to the user predicted by previewAddLiquidity()	✅	❌	✅	-
LIQ-13	Adding liquidity unsafe must provide exact shares to the user predicted by previewAddLiquidity()	✅	❌	✅	-
LIQ-14	Removing liquidity must provide the same amount of base and quote tokens to the user predicted by previewRemoveLiquidity()	✅	✅	✅	65M+
RES-01	If the quote reserve and base reserve of a pool is 0, then the lp total supply must be 0	✅	✅	✅	25M+
RES-02	The base reserve of a pool is always less than or equal to the pool base balance	✅	✅	✅	25M+
RES-03	The quote reserve of a pool is always less than or equal to the pool quote balance	✅	✅	✅	25M+

Invariants Assessed

ID	Description	Tested	Passed	Remediation	Run Count
POOL-01	The sum of the LP token balances held by each user is always equal to the total supply of LP tokens	✅	✅	✅	65M+
POOL-02	sync() must never revert	✅	❌	✅	-
POOL-03	correctRState() must never revert	✅	✅	✅	25M+
POOL-04	The total supply of LP tokens is either 0 or always greater or equal to 1001	✅	✅	✅	25M+
SWAP-01	Swap must decrease the input token balance of the user if the input and output token are different	✅	✅	✅	25M+
SWAP-02	Swap must increase the output token balance of the user if the input and output token are different	✅	✅	✅	25M+
SWAP-03	The swap must credit the user with an amount of the output token that is equal to or greater than the specified minimumOut	✅	✅	✅	25M+

Findings & Resolutions

ID	Title	Category	Severity	Status
C-01	Claiming Yield DoS	DoS	● Critical	Resolved
H-01	Weth Transferred From The Wrong Address	Logical Error	● High	Resolved
H-02	twapUpdate Overflow DoS	DoS	● High	Resolved
H-03	previewAddLiquidity Incorrect quoteBalance	Logical Error	● High	Resolved
H-04	Lacking Gas Yields Claiming Logic	Logical Error	● High	Partially Resolved
H-05	Native Yield Token Yields Cannot Be Configured After Deployment	DoS	● High	Resolved
M-01	Risk Of Function Selector And Storage Collision	Best Practices	● Medium	Acknowledged
M-02	Predictable MagicLP Salt	Gaming	● Medium	Resolved
M-03	previewAddLiquidity Disagrees With _adjustLiquidity	Logical Error	● Medium	Resolved
M-04	Blast Point Remunerations May Be Gamed	Gaming	● Medium	Resolved
M-05	Disabled Native Yield Tokens Lead To Loss Of Funds	Unexpected Behavior	● Medium	Acknowledged
M-06	Lacking LP Validations Allows For Malicious Intent	Validation	● Medium	Resolved
M-07	Governor Contract Does Not Configure Blast Points	Unexpected Behavior	● Medium	Resolved

Findings & Resolutions

ID	Title	Category	Severity	Status
L-01	Unable To Maximize Gas Yields	Optimization	● Low	Resolved
L-02	Typo	Typo	● Low	Resolved
L-03	Lacking Zero Address Validation	Best Practices	● Low	Resolved
L-04	Potentially Unexpected Pool Creator Recorded	Unexpected Behavior	● Low	Resolved
L-05	Lacking SafeCast	Validation	● Low	Resolved
L-06	Quote Target Rounded To 0	DoS	● Low	Resolved
L-07	previewAddLiquidity Can Give Inaccurate Results	Unexpected Behavior	● Low	Resolved
L-08	System Incompatible With Esoteric Token Pairs	Documentation	● Low	Resolved
L-09	Self-Governed Contracts Cannot Change Configuration	Documentation	● Low	Resolved
L-10	Potential Read-Only Reentrancy	Reentrancy	● Low	Resolved
L-11	BlastCauldron Master Contract Can Be Initialized	Unexpected Behavior	● Low	Resolved
L-12	BlastOnboarding DoS For Max Transfer Tokens	DoS	● Low	Acknowledged
L-13	Blast Points Address Configured For Testnet	Warning	● Low	Resolved

Findings & Resolutions

ID	Title	Category	Severity	Status
L-14	Potential Reentrancy Risk	Reentrancy	● Low	Acknowledged
L-15	BlastCauldronV4 Deployment Bricked	DoS	● Low	Acknowledged

C-01 | Claiming Yield DoS

Category	Severity	Location	Status
DoS	● Critical	Global	Resolved

Description

The BlastMagicLP and BlastOnboarding contracts do not expect to hold native Ether and therefore will not accrue claimable native ether yield, however they still attempt to claim native yield with the BlastYields.claimAllNativeYields function.

The implementation of the BLAST_YIELD contract in go reverts if there is not any claimable Ether: https://github.com/blast-io/blast/blob/c39cdf1fa7ef9e0d4eaf64a7a5cf7b3c46c739fd/blast-geth/core/vm/contracts.go#L1239

Recommendation

Either do not invoke the claimAllNativeYields function if there is no native yield to be claimed, or separate the yield claiming logic into independent functions for each type of yield.

Resolution

Abracadabra Team: Resolved.

H-01 | Weth Transferred From The Wrong Address

Category	Severity	Location	Status
Logical Error	● High	Router.sol: 78	Resolved

Description

In the `Router.createPoolETH` function, a deposit to the weth contract is made but then a subsequent weth `safeTransferFrom` is made from the `msg.sender` to the newly created pool.

As a result, users may accidentally pay twice for the WETH the pool should be created with if they had approved the Router contract. The native Ether sent to the Router will be lost.

Recommendation

Use the weth `safeTransferFrom` to transfer from the `Router` contract to the newly created pool rather than from the `msg.sender`.

Resolution

Abracadabra Team: Resolved.

H-02 | twapUpdate Overflow DoS

Category	Severity	Location	Status
DoS	● High	MagicLP.sol	Resolved

Description

In MagicLP.sol, _twapUpdate is called every time setReserve or sync is called. The update adds to _BASE_PRICE_CUMULATIVE_LAST_ which is an ever increasing value.
In DODO v2, because they use an older version of solidity, it is desired and expected for this value to overflow once it has hit the max of type.uint256.

However for MagicLP which uses a newer solidity version, this cannot overflow and all contract functionality will be bricked.

Recommendation

Wrap with an unchecked block to allow for overflow.

Resolution

Abracadabra Team: Resolved.

H-03 | previewAddLiquidity Incorrect quoteBalance

Category	Severity	Location	Status
Logical Error	● High	Router.sol: 90	Resolved

Description

In the `previewAddLiquidity` function the `quoteBalance` is intended to account for the current balance of the quote tokens in the LP as well as the new quote tokens which will be added to the LP. However the `quoteBalance` is the balance of quote tokens + the `baseInAmount`:

```
uint256 quoteBalance = IMagicLP(lp)._QUOTE_TOKEN_().balanceOf(address(lp)) + baseInAmount;
```

Recommendation

Correct the `quoteBalance` to be:

```
uint256 quoteBalance = IMagicLP(lp)._QUOTE_TOKEN_().balanceOf(address(lp)) + quoteInAmount;
```

Resolution

Abracadabra Team: Resolved.

H-04 | Lacking Gas Yields Claiming Logic

Category	Severity	Location	Status
Logical Error	● High	Global	Partially Resolved

Description

There are several contracts in the Abracadabra and Mimswap systems that cannot claim gas yields as they accrue. These contracts include:

- MIM
- FeeRateModel and FeeRateImplementation
- BlastTokenRegistry
- SPELL

Some of these contracts will accrue a large amount of gas yields, for instance MIM will accrue gas yields upon every transfer, approval etc… and the FeeRateModel and FeeRateImplementation will accrue gas yields on every swap in the Mimswap system.

Recommendation

Consider implementing gas yield claiming logic for these contracts.

Resolution

Abracadabra Team: Partially Resolved as MIM will not be redeployed.

H-05 | Native Yield Token Yields Cannot Be Configured After Deployment

Category	Severity	Location	Status
DoS	● High	IBlast.sol: 82	Resolved

Description

In the IERC20Rebasing interface the configure function is defined to return a YieldMode enum value. However the actual native yield precompiles will return a uint256 representing the balance of the user: ERC20Rebasing.

As a result any call to the BlastYields.enableTokenClaimable function will fail when the contract's balance is nonzero (or above 2 wei) as the result will not correctly correspond to an enum value. Therefore the configuration cannot be updated after deployment, or deployment could even be prevented if a malicious actor sends a few wei of the native yield token to the target address.

Recommendation

Correct the `IERC20Rebasing` interface to return a `uint256` value from the configure function rather than a `YieldMode` enum value.

Resolution

Abracadabra Team: Resolved.

M-01 | Risk Of Function Selector And Storage Collision

Category	Severity	Location	Status
Best Practices	● Medium	BlastOnboarding.sol	Acknowledged

Description

The BlastOnboarding contract is itself a proxy, however it has several declared storage variables and functions. As a result the contract is prone to storage and function selector collision.

Recommendation

Any bootstrapper implementation that is used in the future should be rigorously verified to have no collisions with the existing function selectors or storage slots in the BlastOnboarding contract.

Resolution

Abracadabra Team: Acknowledged.

M-02 | Predictable MagicLP Salt

Category	Severity	Location	Status
Gaming	● Medium	Factory.sol	Resolved

Description

The salt used to deploy the new MagicLP contract in the factory is based upon replicable values that any malicious user may pass in.

This way a malicious actor may observe two transactions in the mempool and intentionally get their transaction ordered between them.
Transaction 1: Create pool at address A
Transaction 2: Send additional funds to pool at address A

The attacker may create the exact same pool at address A and end up with the funds in their pool. Currently there is no high impact risk as the owner of the clone does not have any special privileges, but this may be unexpected for the user creating the pool.

Recommendation

Consider including the msg.sender in the salt for pool creation.

Resolution

Abracadabra Team: Resolved.

M-03 | previewAddLiquidity Disagrees With _adjustLiquidity

Category	Severity	Location	Status
Logical Error	● Medium	Router.sol: 492, 493	Resolved

Description

In the _adjustAddLiquidity function the baseAdjustedInAmount and quoteAdjustedInAmount are adjusted without considering tokens that may be sitting in the MagicLP contract and unaccounted for in the reserves.

Therefore the result from _adjustAddLiquidity contradicts the result retrieved from previewAddLiquidity when there are excess tokens sitting in the MagicLP contract as the previewAddLiquidity function accounts for the current token balance of the lp contract.

As a result in some cases the resulting baseAdjustedInAmount and quoteAdjustedInAmount users would expect to pay using the previewAddLiquidity function will not line up with the baseAdjustedInAmount and quoteAdjustedInAmount that are paid in actuality.

Recommendation

Consider accounting for any additional tokens in the lp contract to match the behavior of the previewAddLiquidity function.

Resolution

Abracadabra Team: previewAddLiquidity has been removed.

M-04 | Blast Point Remunerations May Be Gamed

Category	Severity	Location	Status
Gaming	● Medium	Global	Resolved

Description

In the BlastOnboarding contract users may deposit their token balances and will be remunerated with the points they would have otherwise received if they didn't deposit into the contract. These point remunerations will occur off-chain at undisclosed times.

However if a user is able to predict, recognize a pattern, or guess within a reasonable range when the distributions will occur they could deposit before the distribution and withdraw after the distribution to receive more Blast Points than they otherwise would have by simply depositing into the BlastOnboarding contract for the entire period or simply holding the native yield tokens for the entire period.

The same may occur with a malicious Liquidity Provider for the MagicLP contract.

Recommendation

In the case of the BlastOnboarding contract, consider requiring that users have locked amounts to reward them with point distributions. Otherwise ensure there is no way for users to predict when the off-chain point remuneration will occur.

Resolution

Abracadabra Team: Resolved.

M-05 | Disabled Native Yield Tokens Cause Loss Of Funds

Category	Severity	Location	Status
Unexpected Behavior	● Medium	BlastBox.sol	Acknowledged

Description

If the owner does not want to enable native yield for a token with function setTokenEnabled, that token will have the default mode which is AUTOMATIC for WETH and USDB. When a token is in AUTOMATIC mode, the balance of the token in the contract increases as yield is gained. However, the DegenBox contract is unable to support rebasing tokens:

```
/// @notice The BentoBox is a vault for tokens. The stored tokens can be flash loaned and used in strategies.
/// Yield from this will go to the token depositors.
/// Rebasing tokens ARE NOT supported and WILL cause loss of funds.
/// Any funds transfered directly onto the BentoBox will be lost, use the deposit function instead.
```

As per the comment above, rebasing tokens are not compatible with the DegenBox and will cause loss of funds.

Recommendation

Consider adding feature that allows the owner to set token yield mode to VOID to ensure compatibility when the owner does not want token yield enabled.

Resolution

Abracadabra Team: Acknowledged.

M-06 | Lacking LP Validations Allows For Malicious Intent

Category	Severity	Location	Status
Validation	● Medium	Router.sol	Resolved

Description

Anyone can create a pool with the same base/quote tokens as well as the same params (i, k) as an 'official' MIMSwap pool. An attacker could create such pools to steal liquidity away, or even worse implement such pools with rug pull functions.

Since the Router does not validate the lp address, router functions can be used to interact with these malicious LPs. Attackers may interact with router functions intentionally to populate etherscan with transactions to their malicious pool.

Users who see this may check the pool and see that the base/quote token and other params are correct, and end up interacting with that LP instead of the official one. In fact, there may not even be an official lp yet, i.e. attackers frontrun the creation of the LP once they know the address of the base/quote tokens. This often happens when a memecoin is launched with no frontend or official site stating the correct lp address.

Recommendation

In the Router contract, validate the lp address against the pools mapping in the Factory before allowing the function to continue executing.

Resolution

Abracadabra Team: Resolved.

M-07 | Governor Contract Does Not Configure Blast Points

Category	Severity	Location	Status
Unexpected Behavior	● Medium	BlastGovernor.sol	Resolved

Description

The blast governor contract does not configure blast points, this will cause the contract to miss out on points that can later be used to receive an airdrop of tokens from blast.

“Blast Points are distributed automatically every block to EOAs and smart contracts based on their balance of ETH, WETH, and USDB. Specifically, EOAs and smart contracts earn Points at a rate of 0.06504987 Points/Block/ETH (around 0.03252493376 Points/second/ETH).”

Since the contract is meant to hold eth as it collects yields from other contracts, it will not be able to accrue points based on its eth balance.

Recommendation

Add a call to blast points configure function to allow the governor contract's points to be harvested. Add this to constructor `BlastPoints.configure();`

Resolution

Abracadabra Team: Resolved.

L-01 | Unable To Maximize Gas Yields

Category	Severity	Location	Status
Optimization	● Low	Global	Resolved

Description

In the BlastBox and BlastMagicLP contracts the single claimYields function claims both gas yields as well as ETH, WETH, USDB yields.

However the gas yields take 30-days to reach the maximum yield rate, documented here: Receive and claim gas fees - Blast Developer Documentation

As a result, the protocol may wish to have more granular control over the claiming of yields, so as the let the gas yields continue to accrue at the optimal rate.

Recommendation

Consider separating the `claimYields` function into two functions where one can be used to claim ETH, WETH, and USDB yields while another can be used to claim specifically gas yields.

Resolution

Abracadabra Team: Resolved.

L-02 | Typo

Category	Severity	Location	Status
Typo	● Low	MagicLP.sol: 369	Resolved

Description

In the comment on line 369, the word occurring is misspelled as “occuring”.

Recommendation

Replace “occuring” with occurring.

Resolution

Abracadabra Team: Resolved.

L-03 | Lacking Zero Address Validation

Category	Severity	Location	Status
Best Practices	● Low	BlastGovernor.sol: 38	Resolved

Description

In the setFeeTo function there is no validation that the _feeTo address is nonzero.

Recommendation

Add validation that the feeTo address cannot be assigned to 0.

Resolution

Abracadabra Team: Resolved.

L-04 | Potentially Unexpected Pool Creator Recorded

Category	Severity	Location	Status
Unexpected Behavior	● Low	Factory.sol: 82	Resolved

Description

When deploying a new MagicLP pool through the Router.createPool or Router.createPoolETH functions, the creator of the pool will be recorded as the Router contract, with the pool being added to the userPools mapping entry for the router contract.

This may be unexpected as the user who creates the pool through the router would expect to find the newly created pool in their userPools mapping entry, however this new pool will instead be under the router's userPools mapping entry.

Recommendation

Consider if this is expected behavior or not. If the original creator of the pool ought to be recorded, then consider creating a dedicated function for the Router to be able to pass along the address of the user who created the pool through the Router contract.

Resolution

Abracadabra Team: Resolved.

L-05 | Lacking SafeCast

Category	Severity	Location	Status
Validation	● Low	MagicLP.sol: 356, 405	Resolved

Description

In the buyShares and sellShares functions token amounts are casted to uint112 variables for the BASE_TARGET and QUOTE_TARGET, however this casting should make use of the SafeCastLib as is used in the _sync and _setReserve functions to avoid any potential issues for tokens which have a supply in the quadrillions.

Recommendation

Use the SafeCastLib when casting token amounts to uint112 variables in the buyShares and sellShares functions.

Resolution

Abracadabra Team: Resolved.

L-06 | Quote Target Rounded To 0

Category	Severity	Location	Status
DoS	● Low	MagicLP.sol: 379	Resolved

Description

In the buyShares function the _QUOTE_TARGET_ may be rounded to 0 when the quote token has less decimals than the base token. As a result swapping via sellBase will initially be DoS'ed as it relies upon the _SolveQuadraticFunctionForTrade while using the quote target as V0, since R is initially assigned to 1.

Ultimately _SolveQuadraticFunctionForTrade reverts when V0 is 0, causing the DoS on sellBase. This state can be resolved by either depositing more liquidity or correcting the R state so the formula no longer relies on quote token amounts with the correctRState function.

Recommendation

Consider removing this edge case entirely by reverting when the _QUOTE_TARGET_ rounds to 0 when adding initial liquidity with the buyShares function.

Resolution

Abracadabra Team: Resolved.

L-07 | previewAddLiquidity Can Give Inaccurate Results

Category	Severity	Location	Status
Unexpected Behavior	● Low	Router.sol: 117-128	Resolved

Description

In the previewAddLiquidity function, it is possible for the function to return a baseAdjustedInAmount, quoteAdjustedInAmount, and shares combination that cannot be achieved as a result of rounding when assigning the adjusted amounts. For example:

- totalSupply = 10e18
- baseReserve = 10e18
- quoteReserve = 10e6
- baseInput = 1000000000000000010 (1e18 + 10)
- quoteInput = 1000010 (1e6 + 10, quote token is USDC)
- baseInputRatio = (1e18 + 10) * 1e18 / 10e18 = 1e17 + 1
- quoteInputRatio = (1e6 + 10) * 1e18 / 10e6 = 1e17 + 1e12
- baseAdjustedInAmount = baseInput = 1e18 + 1
- quoteAdjustedInAmount = 10e6 * 1e17+1 / 1e18 = 1000000 (1e6)
- shares = totalSupply * (1e17 + 1) / 1e18 = 1e18 + 10

The new baseInputRatio is now larger than the quoteInputRatio as a result of the re-assignment, therefore when these new adjusted values are used to provide liquidity, the minimum of the two input ratios will be different. The minimum will now be the quoteInputRatio of 1e17 rather than the baseInputRatio of 1e17+1.

As a result the shares that are received from using these adjusted inputs will actually be 1e18, rather than 1e18 + 10. Users who set their minimumShares to the result of the previewAddLiquidity function will have their transactions revert.

Recommendation

Consider whether this inaccuracy is acceptable, if it is be sure to clearly document it for users and integrating protocols. If it is not, consider using mulCeil instead of mulFloor, as this would result in user's overspending by a few wei while minting liquidity in these cases, but maintain the fidelity of the preview.

Resolution

Abracadabra Team: Resolved.

L-08 | System Incompatible With Esoteric Token Pairs

Category	Severity	Location	Status
Documentation	● Low	Global	Resolved

Description

The system represents the target price using 1e18 * quoteToken / baseToken, however when the quoteToken has 1e6 decimals (e.g. usdt) and the baseToken has 1e24 decimals (e.g. YamV2) calculations involving I and even computing I may lead to significant precision loss.

Recommendation

Be sure to document that the system is not designed to be compatible with quoteTokens and baseTokens that have a large decimal difference. Ideally the quoteToken is always the token with higher decimals.

Resolution

Abracadabra Team: Resolved.

L-09 | Self-Governed Contracts Cannot Change Configuration

Category	Severity	Location	Status
Documentation	● Low	Global	Resolved

Description

Upon BlastBox and BlastMagicLP deployment/initialization, the yield is configured to be in claimable mode, as well as the governor for the contract is set to be the contract itself as part of the configureDefaultClaimables function call: governorMap[msg.sender] = governor;

To update the governor or change the yield mode, it would require a call to functions configureContract or configureGovernor on the Blast yield contract, although the above mentioned contracts do not have any methods that support doing so.

Recommendation

Document and be aware that since address(this) is the address of the governor, no further Blast configuration can be performed.

Resolution

Abracadabra Team: Resolved.

L-10 | Potential Read-Only Reentrancy

Category	Severity	Location	Status
Reentrancy	● Low	MagicLP.sol: 224, 228	Resolved

Description

The `getQuoteInput` and `getBaseInput` functions rely on the balance of the `MagicLP` contract, which can be manipulated with the use of the `flashLoan` function.

The `flashLoan` function will pass off control of the current tx to an arbitrary to address after sending tokens, e.g. adjusting the balance, and before the user adequately pays for those tokens.
The same balance reliance exists for the `previewAddLiquidity` and `previewRemoveLiquidity` functions on the Router contract as well.

This poses a potential read-only risk for protocols that may choose to integrate with the `MagicLP` system and rely on the `previewAddLiquidity`, `previewRemoveLiquidity`, `getQuoteInput` and `getBaseInput` functions.

Recommendation

Be sure to carefully document these risks for integrating parties, otherwise explicitly implement the `nonReadReentrant` modifier from the Solady library to remove this attack surface.
https://github.com/Vectorized/solady/blob/ec85d4a731c5f69aaa9a324d673f92dac0c29593/src/utils/ReentrancyGuard.sol#L45

Resolution

Abracadabra Team: Resolved.

L-11 | BlastCauldron Master Contract Can Be Initialized

Category	Severity	Location	Status
Unexpected Behavior	● Low	Router.sol: 62, 79	Resolved

Description

Currently there is no mechanism to prevent the initialization of the BlastCauldron master contract. While this poses no immediate risks, it may yield unexpected edge cases in the future.

Recommendation

Consider implementing validation that does not allow the BlastCauldron master contract to be initialized.

Resolution

Abracadabra Team: Resolved.

L-12 | BlastOnboarding DoS For Max Transfer Tokens

Category	Severity	Location	Status
DoS	● Low	BlastOnBoarding.sol	Acknowledged

Description

Some tokens allow users to transfer their entire balance by specifying type(uint256).max as the transfer amount. In such a case where the first user to deposit provides type(uint256).max as an amount parameter to deposit, the totals mapping entry for that token will become the maximum for the uint256 type and therefore no other users will be allowed to onboard new tokens.

Recommendation

There are no known tokens with this behavior on the Blast network currently, however this should be carefully considered when supporting new tokens for the BlastOnboarding contract.

Resolution

Abracadabra Team: Acknowledged.

L-13 | Blast Points Address Configured For Testnet

Category	Severity	Location	Status
Warning	● Low	BlastPoints.sol	Resolved

Description

In the library BlastPoints, the BLAST_POINTS address is hard coded to the testnet implementation.

```
IBlastPoints public constant BLAST_POINTS =
IBlastPoints(0x2fc95838c71e76ec69ff817983BFf17c710F34E0);
```

Recommendation

Be sure to update this address for mainnet deployment.

Resolution

Abracadabra Team: Resolved.

L-14 | Potential Reentrancy Risk

Category	Severity	Location	Status
Reentrancy	● Low	Router.sol	Acknowledged

Description

In the Router contract, addLiquidityETH function, a refund of unused ETH is done before transferring token and addLiquidity which opens up the possibility of reentrancy attacks.

Recommendation

While no attack path was found, it is best to follow CEI pattern and move the refund of ETH to the last action in the function.

Resolution

Abracadabra Team: Acknowledged.

L-15 | BlastCauldronV4 Deployment Bricked

Category	Severity	Location	Status
DoS	● Low	BlastWrappers.sol: 59	Acknowledged

Description

When a new BlastCauldronV4 is deployed, a malicious actor may front-run the init function and use the cook function to trigger an ACTION_CALL to the Blast yields contract that will configure a malicious governor for the Cauldron. As a result the cauldron deployment will then be unusable as the call to init will revert upon attempting to call BlastYields.configureDefaultClaimables.

Recommendation

Be aware of this risk during deployments and consider always initializing a cauldron in the same transaction in which it is deployed.

Resolution

Abracadabra Team: Acknowledged.

Disclaimer

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Guardian to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Guardian’s position is that each company and individual are responsible for their own due diligence and continuous security. Guardian’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by Guardian is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, 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.

Notice that smart contracts deployed on the blockchain are not resistant from internal/external exploit. Notice that active smart contract owner privileges constitute an elevated impact to any smart contract’s safety and security. Therefore, Guardian does not guarantee the explicit security of the audited smart contract, regardless of the verdict.

About Guardian Audits

Founded in 2022 by DeFi experts, Guardian Audits is a leading audit firm in the DeFi smart contract space. With every audit report, Guardian Audits upholds best-in-class security while achieving our mission to relentlessly secure DeFi.

To learn more, visit https://guardianaudits.com

To view our audit portfolio, visit https://github.com/guardianaudits

To book an audit, message https://t.me/guardianaudits