BAS BitAgent Audit Report

Please Note

1. The analysis of the Severity is purely based on the smart contracts mentioned in the Audit Scope and does not include any other potential contracts deployed by the Owner. No applications or operations were reviewed for Severity. No product code has been reviewed.

2. Due to the time limit, the audit team did not do much in-depth research on the business logic of the project. It is more about discovering issues in the smart contracts themselves.

Audit Period: 2025/04/07 - 2025/04/16 (YYYY/MM/DD)

Overall Risk: Pass

Project Name	BAS BitAgent
Website	https://github.com/unibaseio/bitagent-contract

Smart contracts list:

No.	Contract Name	Link	Verdict	Details
1	MCV2_Bond	https://github.com/unibaseio/bitagent-contract/blob/main/contracts/MCV2_Bond.sol	Pass	[CTL01] [CTL02] [CTL03]
2	MCV2_MultiToken	https://github.com/unibaseio/bitagent-contract/blob/main/contracts/MCV2_MultiToken.sol	Pass	
3	MCV2_Royalty	https://github.com/unibaseio/bitagent-contract/blob/main/contracts/MCV2_Royalty.sol	Low	[L01] [CTL04] [CTL05]
4	MCV2_Token	https://github.com/unibaseio/bitagent-contract/blob/main/contracts/MCV2_Token.sol	Pass	
5	VaultManager	https://github.com/unibaseio/bitagent-contract/blob/main/contracts/VaultManager.sol	Pass	
6	ERC1155Initializable	https://github.com/unibaseio/bitagent-contract/blob/main/contracts/lib/ERC1155Initializable.sol	Pass	

7	ERC20Initializable	https://github.com/unibaseio/bitagent-contract/blob/main/contracts/lib/ERC20Initializable.sol	**Pass**	
8	FullMath	https://github.com/unibaseio/bitagent-contract/blob/main/contracts/lib/FullMath.sol	**Pass**	
9	TickMath	https://github.com/unibaseio/bitagent-contract/blob/main/contracts/lib/TickMath.sol	**Pass**	

Centralization Risks

MCV2_Bond.sol

Function	Role	Permission details
updateGraduationSettings	owner	Ability to update global graduation settings
updateVaultManager	owner	Ability to update vaultManager contract that managers creator/protocol rewards for accumulated swap fees from graduated pools

MCV2_Royalty.sol

Function	Role	Permission details
updateProtocolBeneficiary	owner	Ability to update protocolBeneficiary address that receives the accumulated royalties
updateCreationFee	owner	Ability to update creationFee, which is the native token paid for token creation
updateMaxRoyaltyRange	owner	Ability to update maxRoyaltyRange, which is the maximum allowable royalty percentage that can be set for purchases/sales

[CTL01] updateGraduationSettings lacks input checks	
Contract	MCV2_Bond
Severity Level	Centralization
Description	The updateGraduationSettings lacks input checks, allowing admin to input any arbitrary values to the graduation settings JavaScript `/** * @dev Update pool settings */ function updateGraduationSettings( GraduationSettings memory newSettings_ ) external onlyOwner { graduationSettings = newSettings_; }` This could result in manipulation of graduation settings, namely allowing change of maxSupply, curveSupply, graduateMcap, poolLaunchFee, poolTickSpacing, poolFee and sqrtPriceX96.
Recommendation	Consider zero-value and threshold checks, if not, ensure the owner is completely trusted.

[CTL02] updateVaultManager lacks zero address checks	
Contract	MCV2_Bond
Severity Level	Centralization
Description	The updateVaultManager lacks zero address checks, allowing owner to set a zero-address vaultManager contract, which would DoS core functionalities, such as pool graduation. JavaScript `/** * @dev Update reward manager */`

```
    function updateVaultManager(address newManager) external onlyOwner
{
        vaultManager = VaultManager(newManager);
    }
```

Recommendation	Consider a zero address check for the vaultManager contract update, if not, ensure the owner is completely trusted.

[CTL03] Consider events for setter functions

Contract	MCV2_Bond
Severity Level	**Centralization**
Description	The following sensitive setter functions lacks an explicit event emission when state variables are changed ● MCV2_Bond.updateGraduationSettings ● MCV2_Bond.updateVaultManager
Recommendation	Consider explicit events to efficiently track on-chain sensitive variable changes

[CTL04] updateCreationFee lacks threshold checks

Contract	MCV2_Royalty
Severity Level	**Centralization**
Description	The updateCreationFee lacks threshold checks, allowing owner to set an arbitrarily high creation fee during token creation

JavaScript

```
   function updateCreationFee(uint256 amount) external onlyOwner {
        creationFee = amount;

        emit CreationFeeUpdated(amount);
    }
```

Recommendation	Consider a threshold for the vaultManager contract update, if not, ensure the owner is completely trusted.

[CTL05] Malicious protocolBeneficiary can block graduation	
Contract	MCV2_Royalty
Severity Level	**Centralization**
Description	The owner can set a malicious protocolBeneficiary address where the low level call to transfer the creationFee during token creation can revert, possibly causing a DoS in token creation
Recommendation	Ensure the protocolBeneficiary address set is not malicious and that the owner is completely trusted

[L01] Royalty can be bypassed with minimal token purchase and burn	
Contract	MCV2_Bond, MCV2_Royalty.sol
Severity Level	Low

Description	Currently, there is no minimum tokens for tokensToMint and tokensToBurn during purchases and sale of MCV2 tokens via mint and burn respectively. This means that if a sufficiently low amount of tokens are bought or sold, the _getRoyalty function can round down to zero if (reserveAmount * royaltyRatio) < RATIO_BASE, although gas fees would likely not incentivize this
	
	JavaScript
	`function _getRoyalty(`
	`  uint256 reserveAmount,`
	`  uint16 royaltyRatio`
	`) internal pure returns (uint256) {`
	`  return (reserveAmount * royaltyRatio) / RATIO_BASE;`
	`}`

Recommendation	You can consider enforcing that _getRoyalty will return a value greater than zero to disallow royalty bypass.
Status	Waiting for confirmation from the development team.