Nexus Mutual Sherlock Bug Bounty Cover Annex

The Annex is a supplement to and governed by the Nexus Mutual Sherlock Bug Bounty Cover Terms. Capitalized terms not defined herein shall have the meaning assigned to them in the Sherlock Bug Bounty Cover Terms.

All information below sourced from Mellow Finance's Bug Bounty Program page on Sherlock's Bug Bounty Platform as of 24 September 2025.

Rewards by Threat Level

	Critical	High
Max	$100,000	$25,000

Mellow Finance's Critical Definition

- Definite and significant loss of funds without limitations of external conditions
- Definite and significant freezing of funds for >1 year without limitations of external conditions

Sherlock's Bug Bounty Program General Notes

- Sherlock's Criteria for Issue Validity guide (used in Sherlock audit contests) can be a helpful resource for more context on out-of-scope issues, etc. but nothing in the guide should overrule the definitions above
- A coded Proof of Concept (POC) with instructions to run the POC is required
- If the protocol team has the ability to take measures (upgrade the contract, pause the contract, etc.) against an exploit, the potential damage is limited to a 1-hour exploit period before it is assumed that the protocol team takes measures to prevent further damage

Sherlock's Platform Rules

- Please review the Sherlock Bug Bounty Platform Rules before submitting any vulnerability.

Sherlock's Additional Context for Mellow Finance's Bug Bounty Program

Chains in scope

Ethereum, Arbitrum, Base, Hyperliquid L1, Avalanche, Berachain, BSC, OP Mainnet, Polygon, Sonic, Unichain

Expected tokens

Only whitelisted assets are supported in the system: standard ERC20 tokens, native tokens and stETH.

Nexus Mutual

Trusted protocol roles

The following roles are considered trusted. This means entities assigned to these roles are assumed to act in accordance with protocol safety assumptions, and all values they set are expected to be within safe, bounded, and reviewable ranges.

Trusted roles:

- role / most preferred holder type or contract
- `owner` holders in all contracts (including openzeppelin-contracts/contracts/proxy/transparent
- /ProxyAdmin.sol) (admin, proxy-admin)
- `signer` in Consensus.sol contract (admin)
- keccak256("managers.ShareManager.SET_FLAGS_ROLE")
- keccak256("managers.ShareManager.SET_ACCOUNT_INFO_ROLE")
- keccak256("managers.RiskManager.SET_VAULT_LIMIT_ROLE")
- keccak256("managers.RiskManager.SET_SUBVAULT_LIMIT_ROLE")
- keccak256("managers.RiskManager.ALLOW_SUBVAULT_ASSETS_ROLE")
- keccak256("managers.RiskManager.DISALLOW_SUBVAULT_ASSETS_ROLE")
- keccak256("managers.RiskManager.MODIFY_PENDING_ASSETS_ROLE")
- keccak256("managers.RiskManager.MODIFY_VAULT_BALANCE_ROLE")
- keccak256("managers.RiskManager.MODIFY_SUBVAULT_BALANCE_ROLE")
- keccak256("modules.ShareModule.SET_HOOK_ROLE")
- keccak256("modules.ShareModule.CREATE_QUEUE_ROLE")
- keccak256("modules.ShareModule.SET_QUEUE_STATUS_ROLE")
- keccak256("modules.ShareModule.SET_QUEUE_LIMIT_ROLE")
- keccak256("modules.ShareModule.REMOVE_QUEUE_ROLE")
- keccak256("modules.VaultModule.CREATE_SUBVAULT_ROLE")
- keccak256("modules.VaultModule.DISCONNECT_SUBVAULT_ROLE")
- keccak256("modules.VaultModule.RECONNECT_SUBVAULT_ROLE")
- keccak256("modules.VaultModule.PULL_LIQUIDITY_ROLE")
- keccak256("modules.VaultModule.PUSH_LIQUIDITY_ROLE")
- keccak256("oracles.Oracle.SUBMIT_REPORTS_ROLE")
- keccak256("oracles.Oracle.ACCEPT_REPORT_ROLE")
- keccak256("oracles.Oracle.SET_SECURITY_PARAMS_ROLE")
- keccak256("oracles.Oracle.ADD_SUPPORTED_ASSETS_ROLE")
- keccak256("oracles.Oracle.REMOVE_SUPPORTED_ASSETS_ROLE")
- keccak256("permissions.protocols.SymbioticVerifier.CALLER_ROLE")
- keccak256("permissions.protocols.SymbioticVerifier.MELLOW_VAULT_ROLE")
- keccak256("permissions.protocols.SymbioticVerifier.SYMBIOTIC_FARM_ROLE")
- keccak256("permissions.protocols.SymbioticVerifier.SYMBIOTIC_VAULT_ROLE")
- keccak256("permissions.protocols.EigenLayerVerifier.ASSET_ROLE")
- keccak256("permissions.protocols.EigenLayerVerifier.CALLER_ROLE")
- keccak256("permissions.protocols.EigenLayerVerifier.MELLOW_VAULT_ROLE")
- keccak256("permissions.protocols.EigenLayerVerifier.OPERATOR_ROLE")
- keccak256("permissions.protocols.EigenLayerVerifier.RECEIVER_ROLE")
- keccak256("permissions.protocols.EigenLayerVerifier.STRATEGY_ROLE")
- keccak256("permissions.protocols.ERC20Verifier.ASSET_ROLE")
- keccak256("permissions.protocols.ERC20Verifier.CALLER_ROLE")
- keccak256("permissions.protocols.ERC20Verifier.RECIPIENT_ROLE")
- keccak256("permissions.Verifier.SET_MERKLE_ROOT_ROLE")
- keccak256("permissions.Verifier.CALLER_ROLE")
- keccak256("permissions.Verifier.ALLOW_CALL_ROLE")
- keccak256("permissions.Verifier.DISALLOW_CALL_ROLE")

Additional assumptions:

1. No unbounded arrays or arbitrarily large input values will be accepted. All input parameters are constrained to prevent out-of-gas (OOG) conditions during execution.
2. Lockup durations, Oracle timeouts, and other time-sensitive configuration values will be set within non-griefable and operationally safe bounds.
3. Total fee rates configured in the system (e.g., performance + protocol + deposit + redeem fees) will always remain well below 50%.
4. `queueLimit` value will be configured such that no single operation risks exceeding the block gas limit, even under worst-case execution paths.
5. 5. While not enforced at the contract level, the total number of subvaults per MultiVault is assumed to remain under 100.
6. Only implementations explicitly included within this scope will be deployed through Factory contracts.
7. Only hooks explicitly included within this scope will be used in the system.
8. Only queues explicitly included within this scope will be used in the system.
9. Only vault configurations (Vault.sol and Subvault.sol) explicitly included within this scope will be used in the system.

Offchain mechanisms and procedures

- Off-chain Oracle bot: a dedicated off-chain bot is responsible for periodically collecting LP token prices and submitting them as part of Oracle reports.
- Flashbots Usage: flashbots or other private transaction relays may be used to execute actions performed by trusted actors (e.g., admins, operators), especially when timing, frontrunning protection, or MEV resistance is critical.

Protocol Resources

https://mellowprotocol.notion.site/Flexible-Vaults-Architecture-22f02ad86276803c8fdfc694a0036d98

Assets in Scope

- src
    - factories
        - Factory.sol
    - hooks
        - BasicRedeemHook.sol
        - LidoDepositHook.sol
        - RedirectingDepositHook.sol
    - libraries
        - FenwickTreeLibrary.sol
        - ShareManagerFlagLibrary.sol
        - SlotLibrary.sol
        - TransferLibrary.sol
    - managers
        - BasicShareManager.sol
        - FeeManager.sol

        - RiskManager.sol
        - ShareManager.sol
        - TokenizedShareManager.sol
    - modules
        - ACLModule.sol
        - BaseModule.sol
        - CallModule.sol
        - ShareModule.sol
        - SubvaultModule.sol
        - VaultModule.sol
        - VerifierModule.sol
    - oracles
        - Oracle.sol
    - permissions
        - BitmaskVerifier.sol
        - Consensus.sol
        - MellowACL.sol
        - protocols
            - EigenLayerVerifier.sol
            - ERC20Verifier.sol
            - OwnedCustomVerifier.sol
            - SymbioticVerifier.sol
        - Verifier.sol
    - queues
        - DepositQueue.sol
        - Queue.sol
        - RedeemQueue.sol
        - SignatureDepositQueue.sol
        - SignatureQueue.sol
        - SignatureRedeemQueue.sol
    - vaults
        - Subvault.sol
        - VaultConfigurator.sol
        - Vault.sol