SECURITY ASSESSMENT

Denar dUSD

Smart Contract Review

PREPARED FOR	Denar
PREPARED BY	Hashlabs | hashlabs.ch
REPORT DATE	16 September 2026
SCOPE	src directory | 11 Solidity files | 2,174 lines
NETWORK	Robinhood Chain | Chain ID 4663
STATUS	Final

CONFIDENTIAL

This report covers the code and materials listed in scope. It does not certify that the system is free from defects, and it should be read together with the testing limitations described in the report.

Executive Summary

We found one Medium issue, seven Low issues, and eight Informational observations. The review did not identify a Critical or High issue.

The main finding is in StakedDUSD. The contract works out the permanent seed from the share balance held at 0xdEaD. Since any holder can transfer sdUSD to that address, a holder can increase the amount treated as seed until hasStakers returns false. Reward transfers then stop. During the pilot, the cost of causing this condition may be only a few dUSD.

We did not find a route for an unprivileged account to take protocol funds without the owner being involved. The Low findings are mostly failure recovery and trust boundary issues. A broken USDG vault can become impossible to replace; the treasury can return to bootstrap mode while dUSD is still outstanding; a temporary feed failure can open emergency divestment immediately; and the bot hot key also controls the liquidator allow list.

Several core controls are implemented carefully. Facilitator bucket accounting is consistent, distributions are capped by system backing, keeper swaps have two output bounds and a rolling daily limit, and token and feed decimals are checked at deployment. Those strengths are discussed later in the report.

Finding Summary

Severity	Count
Critical	0
High	0
Medium	1
Low	7
Informational	8

Conclusion

M 01 should be fixed before production if the team considers the remediation necessary. Otherwise, the relevant code and configuration should be updated as part of the next applicable deployment. Following any changes, the repository submodules should be initialized and the complete test suite should be run, including the Robinhood Chain fork tests. Ownership should be transferred to the planned multisignature wallet before the system holds material value, after which the final code and deployment configuration should be submitted for external audit.

Assessment Limitations

The review environment did not have Foundry installed. The lib submodules were empty and the supplied folder was not a Git checkout. We therefore did not compile the contracts or run the unit, invariant, fuzz, or fork tests. The conclusions in this report come from manual review of the supplied source code, documentation, and deployment scripts.

Scope and Methodology

Review Scope

Component	Purpose	Lines
stablecoin/DUSD.sol	ERC 20 token with permit and facilitator bucket based minting	81
stablecoin/DUSDTreasury.sol	USDG and dUSD PSM, SGOV reserve, ERC 4626 vault, harvest, and timelocked governance	857
stablecoin/MorphoAllocator.sol	D3M facilitator for Morpho allocation, interest harvesting, loss reserve, and write off	422
stablecoin/StakedDUSD.sol	ERC 4626 sdUSD vault with linear reward vesting and permanent seed	197
oracles/DenarOracle.sol	Morpho oracle using Chainlink feeds and operational safeguards	131
oracles/DenarTwapOracle.sol	Morpho oracle using Uniswap v3 TWAP and Chainlink feeds	294
periphery/DenarLiquidator.sol	Atomic liquidation executor with allow listed swap routers	137
interfaces	Four minimal interfaces for Chainlink, ERC 20, ERC 8056, and backing sources	55

Excluded Components

The following components were outside the review scope: Morpho Blue, MetaMorpho, AdaptiveCurveIrm, the Uniswap libraries vendored under lib, the bots under ops, and the frontend application.

Review Method

We read each of the 11 Solidity files in scope and checked the implementation against docs and the deployment scripts in script. The review covered access control, facilitator accounting, backing calculations, oracle handling, governance delays, keeper swaps, liquidation execution, and denial of service paths.

Compiler Versions

The oracle and liquidator contracts declare Solidity compiler version ^0.8.19. The dUSD contract suite declares Solidity compiler version ^0.8.26.

Severity Classification

Rating	Definition
Critical	Direct and immediate loss of funds or permanent protocol compromise with broad impact.
High	Material loss or freezing of user funds, or a severe compromise requiring limited conditions.
Medium	Meaningful loss of functionality, griefing, or material security weakness without direct theft.
Low	Limited impact, resilience weakness, or issue that depends on trusted or unlikely conditions.
Informational	Design observation, operational consideration, or defense in depth improvement.

Rating Basis

M 01 is rated Medium because it interrupts reward distribution but does not let the attacker steal funds or permanently lock deposits. In a low liquidity pilot, the same issue could reasonably be rated High: the attack is cheap and the failure would be immediately visible to users.

Reference Conventions

File names and line references match the source used for the 16 September 2026 assessment. They may move as the code is changed. Informational items record operational concerns and hardening opportunities; they are not presented as exploitable vulnerabilities.

Findings Overview

ID	Severity	Component	Finding
M 01	Medium	StakedDUSD.sol	Transfers to the burn address can permanently block reward distributions
L 01	Low	DUSDTreasury.sol	A reverting USDG vault cannot be replaced and can block redemptions
L 02	Low	DUSDTreasury.sol	Treasury timelocks can switch off while dUSD remains outstanding
L 03	Low	DUSDTreasury.sol	A transient feed failure can immediately enable emergency divestment
L 04	Low	DenarLiquidator.sol	The bot hot key controls the liquidator allow list and withdrawals
L 05	Low	DUSDTreasury.sol	Corporate action handling can block divestment and depends on issuer views
L 06	Low	DUSDTreasury.sol	Mint and redeem lack output protection while fees may change immediately
L 07	Low	Oracles and Treasury	Chainlink minAnswer and maxAnswer bounds are not checked
I 01	Informational	Multiple contracts	Ownership transfers complete in one step
I 02	Informational	StakedDUSD.sol	A zero vesting period is accepted
I 03	Informational	Treasury and Allocator	Loops over backing sources and markets are unbounded
I 04	Informational	Treasury and Allocator	Harvest may distribute unrealized gains before bad debt recognition
I 05	Informational	DenarOracle.sol	Scheduled multiplier changes are not checked by the market oracle
I 06	Informational	DenarLiquidator.sol	Liquidation callback behavior creates operational edge cases
I 07	Informational	DenarOracle.sol	An issuer pause can freeze a market indefinitely
I 08	Informational	Review environment	Compilation and automated verification were not executed

Detailed Findings

M 01 Transfers to the Burn Address Can Permanently Block Reward Distributions

Severity Medium Location StakedDUSD.sol lines 113 to 123 and 141 to 151

Description

StakedDUSD does not store the number of shares minted during seed(). Instead, it treats the current balance at SEED_RECEIVER as the seed. SEED_RECEIVER is 0x000000000000000000000000000000000000dEaD, and sdUSD remains a normal transferable ERC 20. Any holder can therefore send shares to that address and increase the value the contract regards as seed.

```
function stakedSupply() public view returns (uint256) {
    return totalSupply() - balanceOf(SEED_RECEIVER);
}

function hasStakers() public view returns (bool) {
    if (stakedSupply() < MIN_SHARES) return false;
    return balanceOf(SEED_RECEIVER) * BPS <= totalSupply() * MAX_SEED_BPS;
}
```

Impact

Once the balance at 0xdEaD is more than 5 percent of total supply, hasStakers returns false. transferInRewards then reverts with NoStakers. This also makes DUSDTreasury.harvest revert, while MorphoAllocator keeps the interest as ExcessRetained. No one can move the shares back from 0xdEaD, and there is no owner override. New deposits can dilute the seed ratio below 5 percent, but the attacker can burn more shares and restore the block.

The attacker must give up roughly 5 percent of sdUSD TVL and receives no direct financial benefit. Deposits remain backed and redeemable, and stakers can withdraw rewards that have already vested. The impact is nevertheless material because one public transaction can stop the protocol yield path. With a 1 dUSD seed and a 10,000 dUSD pilot cap, the attack may cost only a few dUSD.

Illustrative Scenario

1. A user deposits 100 dUSD into sdUSD and receives 100 shares.

2. The user transfers the 100 shares to the burn address without requiring any privileged role.

3. The burn address balance exceeds the configured percentage of total supply, causing hasStakers to return false.

4. Subsequent reward distributions revert until honest deposits increase total supply sufficiently or a corrected vault is deployed.

Recommendation

Store the number of shares minted by seed() and use that stored value in stakedSupply and hasStakers. Do not infer the seed from a live token balance. It would also be reasonable to reject transfers to SEED_RECEIVER unless they originate from the seeding flow. Add a regression test that sends sdUSD to 0xdEaD and verifies that harvest still succeeds.

L 01 A Reverting USDG Vault Cannot Be Replaced and Can Block Redemptions

Severity Low Location DUSDTreasury.sol lines 454, 509 to 521, and 543 to 546

Description

_setUsdgVault calls vaultAssets before it replaces the current vault. That check is meant to prevent the owner from abandoning a live position. It also means that a vault which stops answering convertToAssets cannot be replaced or cleared, because the safety check itself reverts.

Impact

The same condition breaks nav, harvest, redeemable, depositToVault, and redeemAllFromVault. A redemption that needs more than the liquid USDG buffer also fails when redeem calls maxWithdraw. In practice, one failing dependency can disable both treasury reporting and the only recovery path for that dependency.

Recommendation

Wrap the current-vault read in try catch inside _setUsdgVault. If the vault cannot report a position, allow the owner to replace it through the existing timelocked path. redeem should also catch failures from maxWithdraw and withdraw, so users can still redeem against USDG already held by the treasury.

L 02 Treasury Timelocks Can Switch Off While dUSD Remains Outstanding

Severity Low Location DUSDTreasury.sol lines 262 to 267, 346 to 349, and 442 to 444

Description

_bootstrapping checks liability() == 0. liability, however, is only the level of the treasury facilitator bucket. Because all dUSD is fungible and redeem burns against that bucket, a user can redeem dUSD minted by the allocator and take the treasury bucket back to zero while other dUSD is still outstanding.

Impact

At that point setSwapTarget and setUsdgVault become immediate owner actions again. The treasury may still hold SGOV, USDG, or vault shares, and the allocator may still have an open dUSD bucket. The timelock therefore depends on the balance of one bucket rather than the state it is intended to protect.

Recommendation

Base bootstrap mode on system state. DUSD_TOKEN.totalSupply() == 0 is the clearest condition; the check may also require zero SGOV and zero vault shares. Since the permanent seed keeps total supply above zero after launch, the timelock would remain active for the life of the deployed system.

L 03 A Transient Feed Failure Can Immediately Enable Emergency Divestment

Severity Low Location DUSDTreasury.sol lines 466 to 473 and 647 to 657

Description

feedIsDead waits seven days for a stale price, but not for other feed failures. A revert, short return value, answer at or below zero, or zero updatedAt makes the feed dead immediately. A single malformed round can therefore open emergencyDivest in the same block.

Impact

emergencyDivest accepts any owner-selected floor of at least 1. If the owner key is compromised, or the operator reacts too quickly to a temporary feed problem, the SGOV position can be sold through an approved router without the normal slippage bound or rolling rotation cap.

Recommendation

Use one persistence rule for every kind of feed failure. For example, store the last time a valid answer was observed and open emergencyDivest only after that timestamp is seven days old. An alternative is a separate declareFeedDead action governed by the treasury timelock.

L 04 The Bot Hot Key Controls the Liquidator Allow List and Withdrawals

Severity Low Location DenarLiquidator.sol lines 19 to 25, 56 to 60, 72 to 81, and 119 to 123

Description

The router allow list is described as protection against a compromised bot. In the documented deployment, however, BOT_OWNER becomes the contract owner. The bot key can call liquidate, change the router allow list, withdraw assets, and transfer ownership.

Impact

Compromising that one key is enough to approve an arbitrary target and remove the liquidator float or collateral in flight. Sweeping balances after each run limits the normal exposure, but the allow list is not a separate layer of protection when the bot itself administers it.

Recommendation

Give the bot an operator role that can only execute liquidations. Keep router approvals, withdrawals, and ownership changes under a multisignature owner. Update the deployment script and runbook so they do not assign both roles to BOT_OWNER.

L 05 Corporate Action Handling Can Block Divestment and Depends on Issuer Views

Severity Low Location DUSDTreasury.sol lines 586 to 589 and 627 to 633

Description

divest raises its price floor when newUIMultiplier is above uiMultiplier. That adjustment fits a dividend reinvestment: the market price is ex dividend and the multiplier restores the missing value. A forward split produces the same multiplier change without increasing the price of each raw token. The floor is then too high and divest can remain unusable for the whole corporate action window.

There is a second dependency in the same area. invest, divest, harvest, and corporateActionPending call oraclePaused, uiMultiplier, and newUIMultiplier directly on the upgradeable SGOV token. If a later token upgrade removes or changes one of those views, every keeper path can revert. emergencyDivest would still be closed while the Chainlink feed remains live.

Recommendation

Apply the floor adjustment only within a plausible dividend range, for example a multiplier ratio no greater than 1.02. Treat larger changes as a corporate action that pauses divestment. Read the token flags through staticcall and allow the explicit-floor emergency path if the expected views stop responding.

L 06 Mint and Redeem Lack Output Protection While Fees May Change Immediately

Severity Low Location DUSDTreasury.sol lines 311 to 328 and 479 to 525

Description

mint and redeem have no minOut or deadline. At the same time, the owner can change tinBps and toutBps immediately, up to 9,999 basis points. A transaction signed against one quoted fee can therefore execute after the owner has changed it.

Impact

The user may receive far less dUSD or USDG than the interface showed. Robinhood Chain has no public mempool, but first come first served ordering does not stop the owner from placing the fee update first. The risk is limited to owner behavior, although it affects the protocol main entry and exit functions.

Recommendation

Add minOut to mint and redeem, with a deadline if the product needs one. The frontend can calculate the expected output and pass the user-selected tolerance. Timelocking fee changes would provide an additional safeguard.

L 07 Chainlink Answer Bounds Are Not Checked

Severity Low Location DenarOracle.sol lines 125 to 130, DenarTwapOracle.sol lines 276 to 281, and DUSDTreasury.sol lines 840 to 845

Description

A Chainlink aggregator may return minAnswer or maxAnswer instead of the true market price once the price leaves its configured range. The reviewed code checks that answers are positive and recent, but it does not detect an answer pinned to either bound.

Impact

If a stock falls below minAnswer, the protocol may keep valuing the collateral at the floor even though the market price is lower. Borrowing could continue against an overstated value. Many newer feeds effectively disable these bounds, but we did not verify the aggregator configuration for each Robinhood Chain stock feed.

Recommendation

Check minAnswer and maxAnswer on the underlying aggregator during deployment. If the feed exposes meaningful limits, store them and reject answers equal to either limit. Otherwise, record the verified configuration for every supported feed in the deployment evidence.

Informational Observations

I 01 Ownership Transfers Complete in One Step

DUSD, DUSDTreasury, MorphoAllocator, StakedDUSD, and DenarLiquidator all use a one-step setOwner call. A mistyped address would permanently lose governance access and, in the treasury, clear queued proposals. Replace this with a transfer and acceptance flow.

I 02 A Zero Vesting Period Is Accepted

StakedDUSD permits vestingPeriod to be zero. Rewards then enter totalAssets immediately and the gradual release no longer protects against deposit and withdrawal around a distribution. The deployment script uses seven days; the constructor should still reject zero.

I 03 Loops Over Backing Sources and Markets Are Unbounded

The treasury loops over every backing source, and the allocator loops over every configured market. Each item requires an external read; the allocator also simulates interest accrual. If harvest runs out of gas, backing may be undercounted and the distribution reduced or blocked. Set explicit list limits and document the gas requirement for keeper calls.

I 04 Harvest May Distribute Unrealized Gains Before Bad Debt Recognition

Morpho positions are counted at par until a liquidation exhausts the collateral and bad debt is recognized. Between a sharp price move and the end of liquidations, harvest may distribute surplus that has already disappeared economically. The equity buffer and vault haircut help, but operations should pause harvest while monitoring shows positions below the agreed threshold.

I 05 Scheduled Multiplier Changes Are Not Checked by the Market Oracle

DenarOracle checks oraclePaused but does not compare newUIMultiplier with uiMultiplier, while the treasury checks both. The difference appears intentional: price per raw token remains continuous through a split or reinvestment, and the ex-dividend window is conservative for collateral value. Add a NatSpec comment so future reviewers do not mistake the difference for an omission.

I 06 Liquidation Callback Behavior Creates Operational Edge Cases

The callback approves the spender for all collateral held by the liquidator, not only the collateral received in the current call. Profit includes only the loan token, so leftover collateral is ignored. Routes that refund native currency also revert because the contract has no receive function. The allow list limits direct exposure, but these cases should be covered in the runbook and tests.

I 07 An Issuer Pause Can Freeze a Market Indefinitely

DenarOracle reverts while the issuer oraclePaused flag is set. This intentionally stops withdrawals and liquidations during a corporate action. If the issuer never clears the flag, the immutable oracle and Morpho market cannot recover in place; the protocol would need a new oracle, a new market, and a manual migration. Prepare that playbook and alert on an unusually long pause.

I 08 Compilation and Automated Verification Were Not Executed

Foundry was unavailable, the submodules were empty, and the supplied folder was not a Git checkout. We did not compile the contracts or run the unit, invariant, fuzz, or fork suites. Initialize the repository and run all tests, including the Robinhood Chain fork cases, before the external audit starts.

Controls Reviewed

The controls below were present and internally consistent when read against the reviewed code paths. This is a source review conclusion only; the controls were not exercised in the unavailable test environment.

Control area	Assessment
Facilitator accounting	DUSD mint and burn operations preserve the relationship between facilitator levels and total supply. The uint128 conversion is bounded by the facilitator cap. The allocator maintains facilitator level at or above total principal through allocation, deallocation, and write off operations.
System wide backing limit	Treasury and allocator distributions are bounded by system surplus after the proposed movement. The design prevents a distribution from reducing total system backing below liabilities and avoids double counting Morpho interest.
Governance delay	The treasury includes a one day timelock, a three day execution window, owner epochs that invalidate proposals after an ownership change, feed validation at application time, and restrictions on replacing a vault with an asset denominated position.
Keeper swap controls	Token approvals are limited to the input amount and reset after execution. Output is measured by balance change. Oracle and keeper floors apply concurrently, and the rolling rotation limit constrains cumulative exposure.
Oracle validation	The TWAP oracle rejects zero values, verifies the pool against the canonical factory, and validates the scale exponent at deployment. The treasury constructor validates USDG, SGOV, and feed decimals.
Staked dUSD controls	The vault uses a permanent seed, protects the supply floor through maxRedeem, prevents totalAssets from falling below unvested rewards, and implements additive continuous reward vesting.

Known Design Risks

Risk area	Description
Weekend market gap	Stock price feeds may remain unchanged from Friday through Monday under the 97 hour staleness threshold. A price gap may exceed the configured loan to value ladder. Liquidation performance depends on Rialto and the latency of the liquidation bot.
Redemption at par	The PSM redeems at one dollar while USDG remains available, even if reserves fall below liabilities. The reserve cushions are intended to prevent this state rather than manage redemptions after it occurs.
Issuer dependencies	USDG is valued at one dollar by construction. SGOV is an upgradeable proxy whose behavior includes issuer controlled operational flags.
Concentrated ownership	Each reviewed contract has a single owner. The planned transfer to a multisignature wallet should occur before material total value is accepted.

Remediation Suggestions

1. Remediate M 01 by storing seedShares and add a regression test covering transfers to the burn address.
2. Initialize all submodules and execute the complete unit, invariant, fuzz, and fork test suites.
3. Separate the DenarLiquidator operator role from the multisignature owner before deploying the bot hot key.
4. Address L 01, L 02, and L 03 as a coordinated treasury governance and resilience update.
5. Transfer ownership to a multisignature wallet before accepting material total value.
6. Commission an independent external audit after remediation and provide the auditor with passing test results and deployment configuration evidence.

Review Cutoff

The review covers the code and documentation supplied for the assessment dated 16 September 2026. Recheck every remediation against the final deployment commit. Changes to contract logic, governance, oracle configuration, or external integrations after that commit fall outside this review.