yAudit

April 2026

Prepared for
Alchemix v3

Audited by
HHK
Panda

Alchemix v3 strategies follow-up audit

Smart Contract Security Assessment

Contents

1 Review Summary

1.1 Protocol Overview

Strategies for Alchemix vaults to generate yield.

1.2 Audit Scope

This audit covers 3 smart contracts totaling approximately 300 lines of code across 1 day of review.

```
v3/
├── script/SimMigrateToWstethStrategy.s.sol
├── src/MYTTokenSwapper.sol
└── src/strategies/WstETHEthereumStrategy.sol
```

1.3 Risk Assessment Framework

1.3.1 Severity Classification

Severity	Description	Potential Impact
Critical	Immediate threat to user funds or protocol integrity	Direct loss of funds, protocol compromise
High	Significant security risk requiring urgent attention	Potential fund loss, major functionality disruption
Medium	Important issue that should be addressed	Limited fund risk, functionality concerns
Low	Minor issue with minimal impact	Best practice violations, minor inefficiencies
Undetermined	Findings whose impact could not be fully assessed within the time constraints of the engagement. These issues may range from low to critical severity, and although their exact consequences remain uncertain, they present a sufficient potential risk to warrant attention and remediation.	Varies based on actual severity
Gas	Findings that can improve the gas efficiency of the contracts.	Increased transaction costs
Informational	Code quality and best practice recommendations	Reduced maintainability and readability

Table 1: severity classification

1.4 Key Findings

Breakdown of Finding Impacts

Impact Level	Count
Critical	0
High	0
Medium	0
Low	2
Informational	4

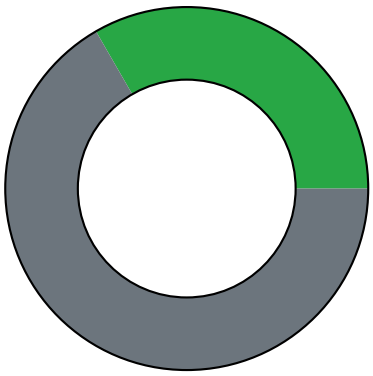

Figure 1: Distribution of security findings by impact level

1.5 Overall Assessment

2 Audit Overview

2.1 Project Information

Protocol Name: Alchemix v3
Repository:
https://github.com/alchemix-finance/v3/tree/2d9f03fa1ed7c209bff779f5b8dfda55ffdca8e7
Commit Hash: 2d9f03fa1ed7c209bff779f5b8dfda55ffdca8e7
Commit URL:
https://github.com/alchemix-finance/v3/tree/2d9f03fa1ed7c209bff779f5b8dfda55ffdca8e7

2.2 Audit Team

HHK, Panda

2.3 Audit Timeline

The audit was conducted from April 23 to 24, 2026.

2.4 Audit Resources

- Code repositories and documentation

2.5 Critical Findings

None.

2.6 High Findings

None.

2.7 Medium Findings

None.

2.8 Low Findings

2.8.1 Oracle failure DoS the entire vault with no clean recovery path

Technical Details

`OraclePricedSwapStrategy._oracleAnswer()` reverts if the Chainlink feed returns a non-positive price, is uninitialized, or is stale (>7 days). This function is called transitively by `_totalValue()`, which is what `MYTStrategy.realAssets()` returns.
Morpho VaultV2 loops every adapter's `realAssets()` inside `accrueInterestView`, which is called at the top of every `accrueInterest()`. The VaultV2 spec explicitly requires (line 119): *"Adapters should not revert on realAssets."* This strategy violates that contract.
When the oracle fails, every user-facing vault function that calls `accrueInterest` is DoS'd: `deposit`, `mint`, `withdraw`, `redeem`, `allocate`. `vault.deallocate` also fails because `MYTStrategy.deallocate` calls `_totalValue()` twice. `forceDeallocate` fails for the same reason plus its internal withdraw.
The only recovery path is `removeAdapter` via timelock. This works but drops `totalAssets` by the full strategy position — socializing a loss equal to the entire wstETH value across all MYT holders. Afterwards, the orphaned wstETH cannot be extracted: vault deallocate is blocked (`isAdapter` check fails), and `rescueTokens` is blocked because wstETH is in `_isProtectedToken`.
The oracle is `immutable`, so there is no in-place fix. The only non-destructive path is waiting for Chainlink to recover.

Impact

Low. Oracle can sometime be deprecated or have issues. Any interruption exceeding the 7-day staleness window freezes the vault. The destructive recovery (remove adapter) realizes a total loss of the strategy's position for all MYT holders, and strands the wstETH with no extraction path.

Recommendation

Make the oracle replaceable (ideally with a short timelock) so the owner can swap to a healthy feed during an outage without removing the adapter. Because `pricedTokenEthOracleDecimals` is also immutable, both must become mutable together (or enforce that replacements match the original decimals).

Developer Response

Fixed in commit f1f2dffd401c3a81ee64be01db9297aae0a2472e.

2.8.2 Migration quote lacks a manual execution-time sanity check

Technical Details

`SimMigrateToWstethStrategy` computes `expectedOut` directly from the Fluid swap target:

```
uint256 expectedOut = swap.getWstETHAmountOut(aWethBal);
uint256 minOut = (expectedOut * SLIPPAGE_NUM) / SLIPPAGE_DEN;
```

The same value is then used to derive `minOut`, so the migration does not independently confirm that the live quote matches an operator-approved value checked shortly before execution.

Impact

Low. This is primarily an operational execution risk. If the quote changes unexpectedly before execution, the script can still proceed as long as the returned value satisfies its internally derived slippage check.

Recommendation

A few minutes before execution, operators should manually fetch and review the expected Fluid output. If the value is acceptable, add that manually approved value to the script and compare it against `expectedOut` immediately after line 142.

```
uint256 manuallyApprovedExpectedOut = XXXXXe18; // Value you got and checked if it makes
 sense.
uint256 maxDeviationBps = 10; // 0.1%

uint256 minApprovedQuote = (manuallyApprovedExpectedOut * (10_000 - maxDeviationBps)) /
10_000;

require(expectedOut >= minApprovedQuote, "Fluid quote below manually approved value");
```

Developer Response

Fixed in commit e5d4bf857d47a1bf9ee871393fe336f816ee923b.

2.9 Gas Savings Findings

None.

2.10 Informational Findings

2.10.1 MYTTokenSwapper migration call lacks a deadline parameter

Technical Details

`MYTTokenSwapper.swapAaveWethToWstethViaFluid` does not accept a deadline parameter, and the migration script (`SimMigrateToWstethStrategy.s.sol`) computes the swap quote and `minAethwstETHOut` ahead of execution at a fixed slippage tolerance:
The migration is multi-signature — between quote computation and on-chain execution, signatures may take hours to collect. If Fluid's pool rate drifts in that window, the trade can either revert (slippage exceeded) or execute at a stale rate that no longer matches operator expectations. The only protection is the static `minAethwstETHOut`, which doesn't enforce any time bound.

Impact

Informational. The slippage tolerance (currently 1bp in the script) provides bounded loss protection but the script uses `getWstETHAmountOut()` for quote so it does not prevent execution against a stale quote. Adding a deadline closes the window cleanly.

Recommendation

Add a `deadline` parameter to `swapAaveWethToWstethViaFluid` and revert if `block.timestamp > deadline`:
Update the migration script to encode a deadline aligned with the expected msig execution window (e.g., `block.timestamp + 2 hours`).

Developer Response

Fixed in commit d89e1c2f0b825c0bb267912c16231fdbf58b0948.

2.10.2 MAX*ORACLE*STALENESS of 7 days is overly permissive for Chainlink stETH/ETH

Technical Details

`OraclePricedSwapStrategy._oracleAnswer()` accepts any non-zero price up to `MAX_ORACLE_STALENESS = 7 days` old. The mainnet Chainlink stETH/ETH feed has a 24-hour heartbeat with a 0.5% deviation trigger — its expected freshness is well under a day. `VaultV2` uses each adapter's `realAssets()` (which calls `_totalValue()` → oracle) inside `accrueInterestView` for share pricing. If the feed stops updating but stays within the 7-day window, deposits/redemptions execute against a stale valuation while real market prices have moved, transferring value between entering and exiting shareholders.

Impact

Informational. Bounded by the stale-price deviation and the strategy's share of vault assets. A 7-day window allows meaningful drift; a heartbeat-aligned bound (~24h) closes most of it.

Recommendation

Tighten the staleness bound to match the deployed feed's heartbeat. Either lower `MAX_ORACLE_STALENESS` globally or make it configurable per-strategy at construction time, so each child sets a value appropriate to its oracle (e.g., 24h for mainnet stETH/ETH, 1h for L2 wstETH/ETH feeds).

Developer Response

Fixed in commit 8247d62a0593390ab7da0cb111fb044cc264a6c2.

2.10.3 Direct WETH-to-wstETH allocation marks down vault shares when stETH trades below ETH

Technical Details

`WstETHEthereumStrategy._allocate(uint256)` mints wstETH at Lido's native 1:1 ETH→stETH rate, but `_positionBalance()` values the result via the stETH/ETH oracle. Whenever the oracle reports stETH below par, the freshly minted position is marked at less than its WETH cost.
The swap path `_allocate(uint256, bytes)` derives `minWstETHOut` from the oracle and slippage tolerance, accepting fills only when the DEX gives back wstETH worth at least the input WETH minus the slippage budget. During a stETH discount, the DEX trades wstETH at the same discount, so the swap path captures more wstETH per ETH and books no immediate loss against the oracle.
Two regimes:

- **Routine drift (~0.1-0.3%)**: Chainlink stETH/ETH typically reports slightly below 1.0. Every direct allocation pays this drag at entry. Lido staking yield (~3-4% APR) recovers it within weeks, so over a sufficient holding period the strategy comes out ahead.
- **Stress events (1-5%+)**: During stETH depegs the entry loss is materially larger. Recovery via yield takes proportionally longer; if the depeg persists or widens, the swap path is strictly better because it captures the discount at entry rather than absorbing it.

There is no clean code-level guard. Adding an oracle floor (`require(ethValue >= amount * (10_000 - slippageBPS) / 10_000)`) would block direct allocation under normal conditions where the oracle sits at 0.999 — leaving capital stuck in idle WETH. The mismatch is structural to oracle-priced ERC4626 vaults that mint at protocol par.

Impact

Informational. The instant accounting loss is real but is offset over time by the underlying Lido yield. The decision between direct and swap is an operational tradeoff: direct captures protocol

staking with a short-term mark-down; swap captures market pricing with no entry loss but consumes DEX liquidity and gas. Allocator role choice determines which is preferable in a given moment.

Recommendation

Treat path selection as an allocator operational discipline rather than a code constraint:

- **During normal conditions** (oracle ~0.999): either path is acceptable. Direct is simpler and avoids DEX dependency; the ~0.1-0.3% mark-down is recovered by yield within weeks.
- **During stETH discount** (oracle < ~0.99): prefer the swap path. It captures the discount at entry rather than locking it in as a strategy mark-down, and avoids the prolonged yield-recovery window.
- **During stETH premium** (oracle > 1.0, rare): direct path is strictly better — it captures the premium as a one-time gain.

Document this as part of the allocator runbook so path selection is deterministic rather than discretionary. Optionally, off-chain monitoring can flag direct allocations executed while the oracle is materially below par for post-hoc review.

Developer Response

Acknowledged. Added to the documentation in commit 16e08825151d80ddda8399c8f52565953c7d4e74.

2.10.4 Natspec missmaatch for `oracleToken()` inside `WstETHEthereumStrategy`

Technical Details

Natspec for the function `oracleToken()` is `/// @notice Returns the token whose amount is priced by the oracle and used in swap sizi` However in the case of `WstETHEthereumStrategy` the variable returned is `wsteth` and not `steth` which is the priced token by the oracle.

Impact

Informational.

Recommendation

Update the natspec to better describe the function and potential overrides.

Developer Response

Fixed in commit fe695fca73708c0ca3ab43716fbcff3e352f94b9

2.11 Final Remarks

The follow-up audit focused on fixing the oracle for the WstEth stategy as well as auditing a script and helper contract to swap aETH into wstETH. The review did not uncover any high severity findings and all issues were promptly addressed by the Alchemix team.

yAudit

Alchemix v3 strategies follow-up audit

Completed 2026-04-24