

Puffer: Puffer Institutional

Security Review

Cantina Managed review by:
Gerard Persoon, Lead Security Researcher
Ladboy233, Security Researcher

June 23, 2026

Contents

1 Introduction **2**
- 1.1 About Cantina . . . 2
- 1.2 Disclaimer . . . 2
- 1.3 Risk assessment . . . 2
  - 1.3.1 Severity Classification . . . 2

2 Security Review Summary **3**
- 2.1 Scope . . . 3

3 Findings **4**
- 3.1 Low Risk . . . 4
  - 3.1.1 An ETH or WETH donation can block `setValidatorsETH()` . . . 4
  - 3.1.2 100% fee cap could lead to division by 0 . . . 4
  - 3.1.3 `getWithdrawalCredentials()` can return `address(0)` . . . 4
  - 3.1.4 Unsafe type conversion . . . 5
  - 3.1.5 `depositScalingFactor` is not explicitly checked . . . 5
  - 3.1.6 Larger `withdrawals[]` arrays will revert . . . 5
  - 3.1.7 Received ETH calculation in `completeQueuedWithdrawals()` might be inaccurate . . . 6
  - 3.1.8 Function `completeQueuedWithdrawals()` might revert if `totalAmount` is larger than `restakedValidatorsETH` . . . 6
  - 3.1.9 Timing of `setValidatorsETH` . . . 6
  - 3.1.10 Low values of `totalAssets()` might get stuck on `require()` in `_accrueFees()` . . . 7
  - 3.1.11 Fee configuration changes can be applied to already-accrued Rewards . . . 8
- 3.2 Gas Optimization . . . 8
  - 3.2.1 For loops with array lengths can be optimized . . . 8
- 3.3 Informational . . . 9
  - 3.3.1 Different ways to send ETH . . . 9
  - 3.3.2 Array lengths not checked . . . 9
  - 3.3.3 There is no function `getLastReportTimestamp()` . . . 9
  - 3.3.4 Could have a constant for `10_000` . . . 10
  - 3.3.5 Error `NotEnoughETH` not accurate . . . 10
  - 3.3.6 Comment for `createVault()` is incorrect . . . 10

1 Introduction

1.1 About Cantina

Cantina is a security services marketplace that connects top security researchers and solutions with clients. Learn more at cantina.xyz

1.2 Disclaimer

Cantina Managed provides a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While Cantina Managed endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that the Cantina Managed security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity level	Impact: High	Impact: Medium	Impact: Low
Likelihood: high	Critical	High	Medium
Likelihood: medium	High	Medium	Low
Likelihood: low	Medium	Low	Low

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. Critical findings have a high likelihood of being exploited and must be addressed immediately. High findings are almost certain to occur, easy to perform, or not easy but highly incentivized thus must be fixed as soon as possible.

Medium findings are conditionally possible or incentivized but are still relatively likely to occur and should be addressed. Low findings are a rare combination of circumstances to exploit, or offer little to no incentive to exploit but are recommended to be addressed.

Lastly, some findings might represent objective improvements that should be addressed but do not impact the project's overall security (Gas and Informational findings).

2 Security Review Summary

Puffer is a decentralized native liquid restaking protocol (nLRP) built on Eigenlayer. It makes native restaking on Eigenlayer more accessible, allowing anyone to run an Ethereum Proof of Stake (PoS) validator while supercharging their rewards.

From Jun 15th to Jun 16th the Cantina team conducted a review of puffer-institutional on commit hash 9dc4b8bf. The team identified a total of **18** issues:

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	0	0	0
Low Risk	11	8	3
Gas Optimizations	1	0	1
Informational	6	6	0
Total	**18**	**14**	**4**

The Cantina Managed team reviewed Puffer Finance's puffer-institutional holistically on commit hash d24a04f2 and concluded that all findings were addressed and no new vulnerabilities were identified.

2.1 Scope

The security review had the following components in scope for puffer-institutional on commit hash 9dc4b8bf:

```
src
├── Errors.sol
├── InstitutionalVault.sol
├── InstitutionalVaultStorage.sol
└── interface
    └── IInstitutionalVault.sol
```

3 Findings

3.1 Low Risk

3.1.1 An ETH or WETH donation can block `setValidatorsETH()`

Severity: Low Risk

Context: InstitutionalVault.sol#L815-L836

Description: An ETH or WETH donation can block `_accrueFees()` and thus `setValidatorsETH()`, which is a known issue.

As we understand from the project one of the reasons for this is to protect incorrect oracle updates, however this check isn't very effective for that.

Recommendation: Consider having a separate sanity check on the values for `restakedValidatorsETH` and `nonRestakedValidatorsETH` to prevent oracle mistakes.

Then in the fee calculation: skip the fee calculation if the reward is outside of the bounds and don't revert.

Puffer: Acknowledged. The breaker is a deliberate halt, not a permanent block. If it trips, the admin clears it by raising the APR cap and reporting, or by removing the donated ETH/WETH via `rescueAnything()` (independent of fee accrual, works for any donation size), so reporting can never be permanently blocked.

We keep the revert rather than skipping the fee, skipping would advance the baseline past a genuine reward and forfeit that fee for good.

Cantina Managed: Acknowledged.

3.1.2 100% fee cap could lead to division by 0

Severity: Low Risk

Context: InstitutionalVault.sol#L58, InstitutionalVault.sol#L861

Description: A 100% fee cap could lead to division by 0.

The code allows:

- `maxTotalFeeBps = 10_000`.
- Sum recipient `bps = 10_000`.

With these values `denom - numByBps` can be zero and the `mulDiv()` of `feeShares` reverts.

Additionally a 100% fee is not practical.

Recommendation: Consider lowering the fee cap and/or check for `denom - numByBps` being zero.

Puffer Finance: Fixed in PR 41.

Cantina Managed: Fix verified.

3.1.3 `getWithdrawalCredentials()` can return `address(0)`

Severity: Low Risk

Context: InstitutionalFactory.sol#L47, InstitutionalVault.sol#L124, InstitutionalVault.sol#L401-L408, InstitutionalVault.sol#L692-L695, InstitutionalVault.sol#L766-L789

Description: If `initializerV3()` isn't run directly after `createVault()`, then `getWithdrawalCredentials()` will return 0 and `startNonRestakingValidators()` uses empty credentials and then funds will not be able to be withdrawn.

Because `createVault()` is authorized the risk of this is low.

Recommendation: Make sure to always call `initializerV3()` directly after `createVault()`.

Also consider adding a check in `getWithdrawalCredentials()` to verify `noRestakingWithdrawalCredentials` is set.

Puffer Finance: Fixed in PR 43.

Cantina Managed: Fix verified.

3.1.4 Unsafe type conversion

Severity: Low Risk

Context: InstitutionalVault.sol#L144, InstitutionalVault.sol#L459, InstitutionalVault.sol#L517-L518, InstitutionalVault.sol#L803-L806, InstitutionalVault.sol#L819, InstitutionalVault.sol#L838, InstitutionalVault.sol#L954-L956

Description: Several unsafe type conversions are done. These could truncate a value if the value is too large.

This won't be an issue in practice because there is not that many ETH. For comparison: `_increaseNetLPFlow()` checks for overflows.

Recommendation: Consider using `SafeCast.toUint128()`.

Puffer Finance: Fixed in PR 42.

Cantina Managed: Fix verified.

3.1.5 `depositScalingFactor` is not explicitly checked

Severity: Low Risk

Context: InstitutionalVault.sol#L443, InstitutionalVault.sol#L455

Description: The `DelegationManager` has a `depositScalingFactor` which could interfere with the value of `scaledShares`.

Recommendation: Consider checking `depositScalingFactor()` has the correct value.

Puffer Finance: Fixed in PR 38.

Cantina Managed: PR 38 reworks `completeQueuedWithdrawals()` to track ETH received via a balance-before/after snapshot rather than deriving amounts from `scaledShares`. As a result, the `depositScalingFactor` no longer affects vault accounting, and no explicit scaling-factor check is needed.

3.1.6 Larger `withdrawals[]` arrays will revert

Severity: Low Risk

Context: InstitutionalVault.sol#L443-L467

Description: The `tokens[]` array is of size 1, but the `withdrawals[]` array could be bigger. In that case `completeQueuedWithdrawals()` will revert because it tries to index `tokens[]` with the same values it indexes `withdrawals[]`.

```
function completeQueuedWithdrawals(...) ... {
       uint256 n = withdrawals.length;
       for (uint256 i; i < n; ++i) {
           _completeQueuedWithdrawal(withdrawals[i], tokens[i], receiveAsTokens[i]);
       }
   }
```

Recommendation: Consider making the array of `tokens[]` the same size as `withdrawals[]`.

Puffer Finance: Fixed in PR 40.

Cantina Managed: Fix verified.

3.1.7 Received ETH calculation in `completeQueuedWithdrawals()` might be inaccurate

Severity: Low Risk

Context: InstitutionalVault.sol#L443-L467, InstitutionalVault.sol#L510-L520

Description: Received ETH calculation in `completeQueuedWithdrawals()` might be inaccurate if slashing has occurred.

Recommendation: Consider using the `balanceBefore` / `balanceAfter` method, similar to `withdrawNonRestakedETH()`.

Puffer Finance: Fixed in PR 38.

Cantina Managed: Fix verified.

3.1.8 Function `completeQueuedWithdrawals()` might revert if `totalAmount` is larger than `restakedValidatorsETH`

Severity: Low Risk

Context: InstitutionalVault.sol#L443-L467, InstitutionalVault.sol#L510-L520

Description: Function `completeQueuedWithdrawals()` might revert if `totalAmount` is larger than `restakedValidatorsETH`. This can happen if the oracle update is not accurate.

Recommendation: Consider using the saturating subtraction like `withdrawNonRestakedETH()` does.

Puffer Finance: Fixed in PR 37.

Cantina Managed: Fix verified.

3.1.9 Timing of `setValidatorsETH`

Severity: Low Risk

Context: InstitutionalVault.sol#L384, InstitutionalVault.sol#L401, InstitutionalVault.sol#L443, InstitutionalVault.sol#L459, InstitutionalVault.sol#L510, InstitutionalVault.sol#L517, InstitutionalVault.sol#L552-L563, InstitutionalVault.sol#L766, InstitutionalVault.sol#L802-L808

Description: The timing of `setValidatorsETH` is important:

- If it is in the same block as a function that increments/decrements `restakedValidatorsETH` / `nonRestakedValidatorsETH`, it will overwrite these values, depending on the order;
- Staking ETH might be in transit for a while and `setValidatorsETH` might not take this into account.

This is also important because the fee calculation is not idempotent (e.g. losses don't have fees, so the order is relevant).

Recommendation: Possible solutions:

- Carefully timing the calls;
- Include the inflight assets in the `setValidatorsETH()` values;
- Revert in `setValidatorsETH` if `restakedValidatorsETH` / `nonRestakedValidatorsETH` are updated in the last N blocks and recalculate the values and try again.

Puffer: Acknowledged. The mitigation is operational that needs to be coordinated with oracle.

Cantina Managed: Acknowledged.

3.1.10 Low values of `totalAssets()` might get stuck on `require()` in `_accrueFees()`

Severity: Low Risk

Context: InstitutionalVault.sol#L124, InstitutionalVault.sol#L144-L146, InstitutionalVault.sol#L815-L823, InstitutionalVault.sol#L815-L842

Description: If `totalAssets()` ever reaches a low value, the `require()` might revert because `rewards` could then be larger than `maxRewards`.

Additionally the early return doesn't do an `emit`.

Recommendation: Consider removing the early return, so the standard `emit` will be used. Only do the require when `totalAssets()` has a non trivial value. For example in the following way:

```
  function _accrueFees(Storage storage $) internal virtual {
  //...
-     if ($.lastTotalAssets == 0) {
-         $.lastTotalAssets = uint128(newTA);
-         $.netLPFlowSinceLastReport = 0;
-         $.lastReportTimestamp = uint64(block.timestamp);
-         return;
-     }

      int256 rewards = ...
      if (rewards > 0) {
          uint256 r = uint256(rewards);
  //...
+         if ($.lastTotalAssets > 1 ether) // otherwise the require might be too strict
              require(r <= maxRewards, RewardsExceedSanityBound());
          _mintFeeShares($, r, newTA);
      }
      $.lastTotalAssets = uint128(newTA);
      $.netLPFlowSinceLastReport = 0;
      $.lastReportTimestamp = uint64(block.timestamp);
      emit FeesAccrued(rewards, newTA);
  }
```

Puffer Finance: Fixed in PR 36.

Cantina Managed: PR 36 addresses this by retaining the early-return guard for the zero-baseline case — removing it would cause `maxRewards` to equal zero (since it scales with `lastTotalAssets`), making any non-zero reward fail the `require()` — and adding a minimum-assets threshold around the sanity check so that the `require()` is only enforced when `lastTotalAssets` exceeds a meaningful value such as 1 ether.

3.1.11 Fee configuration changes can be applied to already-accrued Rewards

Severity: Low Risk

Context: InstitutionalVault.sol#L568-L570

Description: Fee recipient changes are applied to the current unreported reward period, even if the rewards were earned before the change. Fees are only minted when `setValidatorsETH()` is called.

If `10 ETH` of rewards accrued while the recipient list was empty, the admin can set `1000 bps` before the oracle report. The next report mints fee shares using the new `1000 bps`.

As a result, the new recipient receives fees on rewards earned before the fee was configured.

The reverse is also possible: the admin can clear recipients before the report and avoid fees on already-accrued rewards.

Recommendation: Fee configuration changes should only apply to future rewards, not the current unreported period.

Consider the fix: Accrue fees before changing recipients or bps.

Puffer: Acknowledged. Fee configuration (`setFeeRecipients()`, `setMaxTotalFeeBps()`, `setMaxAprBps()`) is restricted to the trusted vault admin, who already holds strictly broader control over share supply and assets via `adminMint()`, `adminBurn()`, and `rescueAnything()`. Any value obtainable by timing a fee-config change around an oracle report is a subset of those existing privileges and grants no additional capability.

Cantina Managed: Acknowledged.

3.2 Gas Optimization

3.2.1 For loops with array lengths can be optimized

Severity: Gas Optimization

Context: InstitutionalVault.sol#L452, InstitutionalVault.sol#L494, InstitutionalVault.sol#L780, InstitutionalVault.sol#L791, NonRestakingWithdrawalCredentials.sol#L72, NonRestakingWithdrawalCredentials.sol#L97

Description: For loops with array lengths can be optimized because in every loop iteration the length is calculated.

Recommendation: Consider caching the array length.

Puffer: Acknowledged. This is a negligible optimization: storage-array loops already cache the length, and for calldata arrays the per-iteration length read is cheap. We leave the loops as-is, also because caching it in startValidators would add a local that might trips a stack-too-deep in that function. Not worth changing.

Cantina Managed: Acknowledged.

3.3 Informational

3.3.1 Different ways to send ETH

Severity: Informational

Context: InstitutionalVault.sol#L229, InstitutionalVault.sol#L250-L251, InstitutionalVault.sol#L294, InstitutionalVault.sol#L315-L316, NonRestakingWithdrawalCredentials.sol#L58-L61

Description: Different ways to send ETH are used:

- `NonRestakingWithdrawalCredentials::withdrawETH()` uses `Address.sendValue()`;
- `InstitutionalVault::redeemETH()` and `InstitutionalVault::withdrawETH()` use `receiver.call{...}()`.

This is inconsistent.

Recommendation: Consider using the same method everywhere.

Puffer Finance: Fixed in PR 39.

Cantina Managed: Fix verified.

3.3.2 Array lengths not checked

Severity: Informational

Context: InstitutionalVault.sol#L443-L467

Description: Array lengths in `completeQueuedWithdrawals()` are not checked:

Recommendation: Consider adding the following checks:

- `withdrawals.length == receiveAsTokens.length`.
- `withdrawals[i].strategies.length == 1`.
- `withdrawals[i].scaledShares.length == 1`.

Puffer Finance: Fixed in PR 40.

Cantina Managed: Fix verified.

3.3.3 There is no function `getLastReportTimestamp()`

Severity: Informational

Context: InstitutionalVault.sol#L726-L761

Description: Several values can be retrieved, except from `lastReportTimestamp`.

Recommendation: Consider adding a function `getLastReportTimestamp()` if this is useful.

Puffer Finance: Fixed in PR 39.

Cantina Managed: Fix verified.

3.3.4 Could have a constant for `10_000`

Severity: Informational

Context: InstitutionalVault.sol#L58, InstitutionalVault.sol#L832, InstitutionalVault.sol#L860

Description: The value of `10_000` is used several times. If this is ever updated, one location could be forgotten.

Recommendation: Consider having a constant for `10_000`.

Puffer Finance: Fixed in PR 41.

Cantina Managed: Fix verified.

3.3.5 Error `NotEnoughETH` not accurate

Severity: Informational

Context: InstitutionalVault.sol#L918-L936

Description: The error `NotEnoughETH` is not accurate, because the check is done for `WETH`.

Recommendation: Consider changing the error in the following way:

```
- NotEnoughETH
+ NotEnoughWETH
```

Puffer Finance: Fixed in PR 39.

Cantina Managed: Fix verified.

3.3.6 Comment for `createVault()` is incorrect

Severity: Informational

Context: InstitutionalFactory.sol#L38-L80

Description: A comment for `createVault()` is incorrect, because the returned values are in the reverse order.

```
@return The address of the vault and the address of the access manager
```

Recommendation: Consider updating the comment and also consider using named return variables.

Puffer Finance: Fixed in PR 39.

Cantina Managed: Fix verified.