GUARDIAN

Alongside

Universal Vault

Security Assessment

July 22nd, 2025

Summary

Audit Firm Guardian

Prepared By Daniel Gelfand, Parker Stevens, Pyro, Arsen, Vladimir Zotov

Client Alongside

Final Report Date July 22, 2025

Audit Summary

Alongside engaged Guardian to review the security of their Universal Vault allowing users to deposit and earn yield from customized strategies. From the 7th of July to the 14th of July, a team of 5 auditors reviewed the source code in scope. All findings have been recorded in the following report.

Confidence Ranking

Given the lack of critical issues detected and minimal code changes following the main review, Guardian assigns a Confidence Ranking of 5 to the protocol. Users should be aware that the protocol relies on multiple trust assumptions related to privileged roles, including the owner, manager, and oracle updater. These roles retain permissions that can significantly affect protocol behavior, and their actions should be considered part of the system's trust model. Furthermore, there should be careful consideration which underlying assets are used for the vaults to prevent unexpected behavior. Guardian advises the protocol to consider periodic review with future changes. For detailed understanding of the Guardian Confidence Ranking, please see the rubric on the following page.

🔗 Blockchain network: **Base, Arbitrum, Katana**

✅ Verify the authenticity of this report on Guardian's GitHub: https://github.com/guardianaudits

📊 Code coverage & PoC test suite: https://github.com/GuardianOrg/universal-vault-fuzz

Guardian Confidence Ranking

Confidence Ranking	Definition and Recommendation	Risk Profile
5: Very High Confidence	Codebase is mature, clean, and secure. No High or Critical vulnerabilities were found. Follows modern best practices with high test coverage and thoughtful design. **Recommendation:** Code is highly secure at time of audit. Low risk of latent critical issues.	0 High/Critical findings and few Low/Medium severity findings.
4: High Confidence	Code is clean, well-structured, and adheres to best practices. Only Low or Medium-severity issues were discovered. Design patterns are sound, and test coverage is reasonable. Small changes, such as modifying rounding logic, may introduce new vulnerabilities and should be carefully reviewed. **Recommendation:** Suitable for deployment after remediations; consider periodic review with changes.	0 High/Critical findings. Varied Low/Medium severity findings.
3: Moderate Confidence	Medium-severity and occasional High-severity issues found. Code is functional, but there are concerning areas (e.g., weak modularity, risky patterns). No critical design flaws, though some patterns could lead to issues in edge cases. **Recommendation:** Address issues thoroughly and consider a targeted follow-up audit depending on code changes.	1 High finding and ≥ 3 Medium. Varied Low severity findings.
2: Low Confidence	Code shows frequent emergence of Critical/High vulnerabilities (~2/week). Audit revealed recurring anti-patterns, weak test coverage, or unclear logic. These characteristics suggest a high likelihood of latent issues. **Recommendation:** Post-audit development and a second audit cycle are strongly advised.	2-4 High/Critical findings per engagement week.
1: Very Low Confidence	Code has systemic issues. Multiple High/Critical findings (≥5/week), poor security posture, and design flaws that introduce compounding risks. Safety cannot be assured. **Recommendation:** Halt deployment and seek a comprehensive re-audit after substantial refactoring.	≥5 High/Critical findings and overall systemic flaws.

Table of Contents

Project Overview

Project Summary

Project Name	Alongside
Language	Solidity
Codebase	https://github.com/Alongside-Finance/universal-vault-contracts
Commit(s)	Initial commit(s): eecb39a11c6939452b72ef75fcff144d82921cd7 Final commit: f10ebecf98327e27dc0d2c37923c59c048153d32

Audit Summary

Delivery Date	July 22, 2025
Audit Methodology	Static Analysis, Manual Review, Test Suite, Contract Fuzzing

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
● Critical	0	0	0	0	0	0
● High	0	0	0	0	0	0
● Medium	2	0	0	0	0	2
● Low	6	0	0	4	0	2
● Info	9	0	0	4	0	5

Audit Scope & Methodology

```
Scope and details:
contract,source,total,comment
universal-vault-contracts/src/WithdrawalQueue.sol,328,557,138
universal-vault-contracts/src/VaultOracle.sol,228,450,153
universal-vault-contracts/src/VaultFactory.sol,88,177,66
universal-vault-contracts/src/UniversalVault.sol,255,620,274
source count: {
  total: 1804,
  source: 899,
  comment: 631,
  single: 65,
  block: 566,
  mixed: 3,
  empty: 277,
  todo: 8,
  blockEmpty: 0,
  commentToSourceRatio: 0.7018909899888766
}
```

Audit Scope & Methodology

Vulnerability Classifications

Severity	Impact: *High*	Impact: *Medium*	Impact: *Low*
Likelihood: *High*	● Critical	● High	● Medium
Likelihood: *Medium*	● High	● Medium	● Low
Likelihood: *Low*	● Medium	● Low	● Low

Impact

High	Significant loss of assets in the protocol, significant harm to a group of users, or a core functionality of the protocol is disrupted.
Medium	A small amount of funds can be lost or ancillary functionality of the protocol is affected. The user or protocol may experience reduced or delayed receipt of intended funds.
Low	Can lead to any unexpected behavior with some of the protocol's functionalities that is notable but does not meet the criteria for a higher severity.

Likelihood

High	The attack is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount gained or the disruption to the protocol.
Medium	An attack vector that is only possible in uncommon cases or requires a large amount of capital to exercise relative to the amount gained or the disruption to the protocol.
Low	Unlikely to ever occur in production.

Audit Scope & Methodology

Methodology

Guardian is the ultimate standard for Smart Contract security. An engagement with Guardian entails the following:

- Two competing teams of Guardian security researchers performing an independent review.
- A dedicated fuzzing engineer to construct a comprehensive stateful fuzzing suite for the project.
- An engagement lead security researcher coordinating the 2 teams, performing their own analysis, relaying findings to the client, and orchestrating the testing/verification efforts.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross-referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts. Comprehensive written tests as a part of a code coverage testing suite.
- Contract fuzzing for increased attack resilience.

Invariants Assessed

During Guardian's review of Alongside, fuzz-testing was performed on the protocol's main functionalities. Given the dynamic interactions and the potential for unforeseen edge cases in the protocol, fuzz-testing was imperative to verify the integrity of several system invariants.

Throughout the engagement the following invariants were assessed for a total of 5,000,000+ runs with a prepared fuzzing suite.

ID	Description	Tested	Passed	Remediation	Run Count
GLOB-01	Vault's maxDeposit() ≈ previewMint(maxMint())	✅	❌	✅	5,000,000+
GLOB-02	Vault's totalAssets() = convertToAssets(totalSupply())	✅	✅	✅	5,000,000+
GLOB-03	Vault's maxMint() = previewDeposit(maxDeposit())	✅	✅	✅	5,000,000+
GLOB-04	maxWithdraw(this) < totalAssets()	✅	✅	✅	5,000,000+
GLOB-05	convertToAssets(maxMint()) < maxDeposit()	✅	✅	✅	5,000,000+
GLOB-06	nextFinalizedWithdrawalId() < nextWithdrawalId()	✅	✅	✅	5,000,000+
GLOB-07	No gaps in completed finalization requests	✅	✅	✅	5,000,000+
GLOB-08	nextCompletedWithdrawId() < nextFinalizedWithdrawalId()	✅	✅	✅	5,000,000+
GLOB-09	Every request's checkpointPtr valid	✅	✅	✅	5,000,000+
GLOB-10	Last checkpoint upperBound +1 = nextCompletedWithdrawId()	✅	✅	✅	5,000,000+

Invariants Assessed

ID	Description	Tested	Passed	Remediation	Run Count
GLOB-11	Every request's checkpointPtr < nextCheckpointId	✅	✅	✅	5,000,000+
GLOB-12	NFTs exist for non-completed withdrawal IDs	✅	✅	✅	5,000,000+
GLOB-13	Active observations length = finalize requests - completes	✅	✅	✅	5,000,000+
GLOB-14	Active list points to valid observations	✅	✅	✅	5,000,000+
GLOB-15	Checkpoints have monotonically increasing contiguous bounds	✅	✅	✅	5,000,000+
GLOB-16	Binary search matches linear search results	✅	✅	✅	5,000,000+
DEP-01	totalAssets() < totalStakedLimit()	✅	✅	✅	5,000,000+
DEP-02	Post-deposit totalAssets < pre + assets (approx eq)	✅	✅	✅	5,000,000+
DEP-03	Post-deposit maxDeposit > pre - assets (approx)	✅	✅	✅	5,000,000+
MNT-01	totalAssets() < totalStakedLimit()	✅	✅	✅	5,000,000+
MNT-02	Post-mint totalSupply = pre + shares	✅	✅	✅	5,000,000+
MNT-03	Post-mint maxMint > pre - shares (approx)	✅	✅	✅	5,000,000+
REQW-01	Current epoch price = 0 post-request withdrawal	✅	✅	✅	5,000,000+
CMPLT-01	IntervalKey not active after completion	✅	✅	✅	5,000,000+

Findings & Resolutions

ID	Title	Category	Severity	Status
M-01	Incorrect maxMint Leads To Mint DoS	Logical Error	● Medium	Resolved
M-02	Incorrect Mint Slippage Protection	Logical Error	● Medium	Resolved
L-01	MAX_PRICE_CHANGE May Be Overly Restrictive	Warning	● Low	Resolved
L-02	New Manager Needs Prior Manager's Assets	Warning	● Low	Acknowledged
L-03	Low Decimal Tokens Unsupported	Warning	● Low	Acknowledged
L-04	Lack Of Minimum Deposit	Warning	● Low	Acknowledged
L-05	maxDeposit Rounds Up	Rounding	● Low	Resolved
L-06	Rebases In Queue Trap Funds	Informational	● Low	Acknowledged
I-01	Modifier Never Used	Best Practices	● Info	Resolved
I-02	Zero Epoch Deposits Panic	Documentation	● Info	Resolved
I-03	Trust Assumptions	Documentation	● Info	Acknowledged
I-04	Unused VaultDeployed Event	Best Practices	● Info	Resolved
I-05	Staked Limit Passed	Warning	● Info	Acknowledged

Findings & Resolutions

ID	Title	Category	Severity	Status
I-06	Pausable Initializer Not Called	Best Practices	● Info	Resolved
I-07	Unused Imports	Best Practices	● Info	Resolved
I-08	Owner Discrepancy	Configuration	● Info	Acknowledged
I-09	Contract Not 4626 Compliant	Best Practices	● Info	Acknowledged

Remediated Findings & Resolutions

ID	Title	Category	Severity	Status
L-01-R	Accumulated Price Can Exceed Interval	Logical Error	● Low	Resolved
I-01-R	Suffix Typo	Informational	● Info	Resolved
I-02-R	Visibility Conventions	Informational	● Info	Resolved
I-03-R	Incorrect Natspec	Informational	● Info	Resolved
I-04-R	Unreachable Code	Informational	● Info	Resolved
I-05-R	No Two Vaults Can Have Same Name And Symbol	Documentation	● Info	Acknowledged
I-06-R	Redundant Activity Check	Documentation	● Info	Resolved
I-07-R	Vault Pause And Activation Assymetry	Documentation	● Info	Acknowledged

M-01 | Incorrect maxMint Leads To Mint DoS

Category	Severity	Location	Status
Logical Error	● Medium	UniversalVault.sol: 427	Resolved

Description

Currently function maxMint returns previewMint on the maxDeposit

```
    function maxMint() public view returns (uint256) {

         return previewMint(maxDeposit());

    }
```

However there is an issue with that as previewMint is meant to accept shares and maxDeposit returns assets:

```
    function previewMint(uint256 _shares) public view returns (uint256) {
         UniversalVaultStorage storage $ = _getUniversalVaultStorage();

         return _convertToAssets(
              _shares, $.oracle.getLatestPrice(address(this)), Math.Rounding.Ceil
         );

    }
```

This would mean that previewMint would convert our assets into assets (thinking they were shares) which would break the whole max assets/shares that anyone is able to deposit.

Depending on the ratio between the two, this can either significantly decrease the cap for which mint can deposit up to or in the more dangerous scenario - significantly increase it.

Recommendation

Change previewMint to previewDeposit within maxMint.

Resolution

Alongside Team: The issue was resolved in commit 9f0ea52.

M-02 | Incorrect Mint Slippage Protection

Category	Severity	Location	Status
Logical Error	● Medium	UniversalVault.sol: 229	Resolved

Description

Function mint(uint256 _shares, address _receiver, uint256 _minAssets) takes a _minAssets parameter as a form of slippage protection:

```
      if (_minAssets = 0 && assets < _minAssets) {

          revert SlippageError(assets, _minAssets);

      }
```

However, for proper slippage control, mint must revert if minting _shares costs more than a maxAssets of underlying tokens, rather than less than minAssets.

This is because if the shares become more expensive, more assets may be required than the user expected.

Recommendation

Change function mint to use maxAssets instead of a _minAssets parameter, and update the inequality.

Resolution

Alongside Team: The issue was resolved in commit 7636b28.

L-01 | MAX_PRICE_CHANGE May Be Overly Restrictive

Category	Severity	Location	Status
Warning	● Low	VaultOracle.sol	Resolved

Description

Function `_checkPriceChange` restricts price changes to 20 bps per update interval (at least 1 hour).

Although this may function normally for most strategies, if the manager is utilizing a strategy that handles external trading positions, a much larger price movement can occur in that time period that may not be appropriately delta neutral.

Afterwards, the price change will be exceeded and the oracle updater will be unable to update the price, progress the epochs forward, and process redemptions.

Recommendation

Consider turning `MAX_PRICE_CHANGE` in a mutable variable that can be updated by the admin.

Resolution

Alongside Team: The issue was resolved in commit 79d1d417.

L-02 | New Manager Needs Prior Manager's Assets

Category	Severity	Location	Status
Warning	● Low	Global	Acknowledged

Description

Appointing a new vault manager does not automatically transfer funds from the old manager, potentially leaving the new manager without assets to process withdrawals and trapping users' funds.

The new manager would need control of the assets as well as ownership of any external positions.

Recommendation

Ensure all necessary assets are transferred to the new manager.

Resolution

Alongside Team: Acknowledged, this was already considered.

L-03 | Low Decimal Tokens Unsupported

Category	Severity	Location	Status
Warning	● Low	VaultOracle.sol: 479	Acknowledged

Description

Although the Universal Vault will be primarily used for universal assets, the Vault was designed to be token-agnostic.

However, low decimal tokens may suffer excessive precision loss (e.g., 2 decimals like GUSD), such that oracle price updates are limited or entirely prevented.

In the case of GUSD, `maxDiff` would floor to 0 and DoS Oracle/Queue operations: `uint256 maxDiff = (lastPrice * MAX_PRICE_CHANGE) / 1e18;`

Recommendation

Carefully select which assets will be used with the Vault and document this risk.

Resolution

Alongside Team: Acknowledged. We are going to use these vaults with uAssets (standard ERC20 tokens with 18 decimals) only. We are also considering USDC in the future, but not confirmed.

L-04 | Lack Of Minimum Deposit

Category	Severity	Location	Status
Warning	● Low	UniversalVault.sol: 201, 229	Acknowledged

Description

Currently there is a minimum withdrawal amount but not a minimum deposit amount. Non-malicious users typically do not deposit a couple wei of assets, and having that symmetrical validation would help ensure a user does not deposit and become stuck instantaneously.

Recommendation

Consider adding a minimum deposit amount of assets, or clearly document this behavior.

Resolution

Alongside Team: Acknowledged. This behavior is intended, we let users to deposit any amount as long as shares are not zero. Therefore, users have two options, wait until their invest have surpassed the threshold or deposit more uAssets, since the restriction to withdraw was implemented merely to prevent from spamming attacks in the WithdrawalQueue. This might be considered in the future.

L-05 | maxDeposit Rounds Up

Category	Severity	Location	Status
Rounding	● Low	src/UniversalVault.sol	Resolved

Description

maxDeposit is calculated as totalStakedLimit() - totalAssets(), where totalAssets() is calculated with FLOOR rounding. Because totalAssets is used to subtract from totalStakedLimit, but totalAssets is rounded down, this would increase the overall value of the maxDeposit.

Recommendation

Note that behaviour and set the limit accordingly, or roundup stakedAssets specifically within function maxDeposit().

Resolution

Alongside Team: The issue was resolved in commit e930413.

L-06 | Rebases In Queue Trap Funds

Category	Severity	Location	Status
Informational	● Low	src/WithdrawalQueue.sol	Acknowledged

Description

The vaults should be agnostic and be able to use any tokens, however if used with rebasing tokens the withdraw queue will experience rebases inside of it, as there would be time gaps between the manager calling `completeFinalizeWithdraw` and all users collecting their withdraws with `claimWithdraw`.

During those time gaps rebases may occur, which would result in that rebase being bricked inside the contract.

If negative rebases occur the manager may be required to separately send extra tokens in order to allow for all users to withdraw.

Recommendation

It's not recommended to use tokens such as `stETH` or any other rebasing tokens.

Resolution

Alongside Team: Acknowledged. We are going to use these vaults with uAssets (standard ERC20 tokens with 18 decimals) only. We are also considering USDC in the future, but not confirmed.

I-01 | Modifier Never Used

Category	Severity	Location	Status
Best Practices	● Info	WithdrawalQueue.sol: 123	Resolved

Description

Modifier `onlyVaultOwner` is defined but never used within the `WithdrawalQueue`.

Recommendation

Consider removing the extraneous modifier definition.

Resolution

Alongside Team: The issue was resolved in commit de78e5d.

I-02 | Zero Epoch Deposits Panic

Category	Severity	Location	Status
Documentation	● Info	Global	Resolved

Description

If a Vault is deployed without being activated in the `VaultOracle` in-tandem, user deposits will panic underflow when calling `getLatestPrice` since the epoch will be 0 for the vault:

```
$.prices[vaultAddr][$.vaults[vaultAddr].epoch - 1];
```

This may be an unexpected error and a more verbose custom error may be preferred.

Recommendation

Clearly document this or consider adding a more verbose error such as '`VaultNotActiveYet`'.

Resolution

Alongside Team: The issue was resolved in commit 62836ff.

I-03 | Trust Assumptions

Category	Severity	Location	Status
Documentation	● Info	Global	Acknowledged

Description

Users of the Universal Vault system must trust a set privileged actors to act in good faith, including but not limited to:

Manager Trust Assumptions:
• Securely manages and protects user deposited assets.
• Uses assets appropriately in yield-generating strategies.
• Approves an allowance to the `WithdrawalQueue` and returns assets when users request withdrawals.
• Requests withdraw finalization and completes batches in a timely, proper manner.

Oracle Updater Trust Assumption:
• Accurately prices in each epoch and handle price volatility.
• Updates prices in a timely manner to ensure smooth withdrawal flow and without excess gas usage on claim.

Vault Owner Trust Assumptions:
• Pauses the Vault when necessary
• Sets parameters such that the Vault operates safely, e.g. enforcing a large enough `minWithdraw` to prevent spam and likely malicious withdrawal requests.

Recommendation

Clearly document trust assumptions for privileged actors as well as the specs for the offchain system.

Resolution

Alongside Team: Acknowledged. Everything mentioned here will be properly documented prior to Vault's launch.

I-04 | Unused VaultDeployed Event

Category	Severity	Location	Status
Best Practices	● Info	VaultFactory.sol	Resolved

Description

Although `event VaultDeployed` is defined it is not used which may negatively impact frontends relying on these emitted events.

Recommendation

Emit the event within function `deployVault`.

Resolution

Alongside Team: The issue was resolved in commit 12d0f04.

I-05 | Staked Limit Passed

Category	Severity	Location	Status
Warning	● Info	UniversalVault.sol	Acknowledged

Description

The totalStakedLimit does not account for pending withdrawal assets, which are still held by the manager post-share burn but pre-finalization, allowing new deposits to exceed the effective limit and potentially overcommitting the strategy.

Although this may be the intended behavior by the Vault Owner and Manager, it should be clearly documented.

Recommendation

Clearly document this behavior.

Resolution

Alongside Team: Acknowledged. As you mentioned, this is intended as we don't consider those pending withdrawals as regular positions since at some point they will be withdrawn but there will be a period in between where those “positions” will remain exposed to loses in the vault. We will documents this behavior more explicitly.

I-06 | Pausable Initializer Not Called

Category	Severity	Location	Status
Best Practices	● Info	UniversalVault.sol: 114	Resolved

Description

Function `__Pausable_init()` is not called within the `initialize` function which goes against best practices to call `__{ContractName}_init` functions for all directly inherited contracts.

Recommendation

Consider adding `__Pausable_init()` in the `initialize` function.

Resolution

Alongside Team: The issue was resolved in commit 6b12a9a.

I-07 | Unused Imports

Category	Severity	Location	Status
Best Practices	● Info	Global	Resolved

Description

• IERC20Metadata is imported within the UniversalVault but never used.

• Math is imported within the Withdrawal Queue but never used.

Recommendation

Consider removing the unused import.

Resolution

Alongside Team: The issue was resolved in commit 3504d5d.

I-08 | Owner Discrepancy

Category	Severity	Location	Status
Configuration	● Info	VaultFactory.sol: 168	Acknowledged

Description

When the factory initializes a UniversalVault and Withdrawal Queue, it sets the owner of these entities as the owner() of the Factory itself.

However, if the Factory updates its owner via Ownable2Step, it doesn't update the owner for previously initialized Vaults and Withdrawal Queues.

This creates a situation where outdated/incorrect owners exist for entities created via the factory.

Recommendation

Be aware of this scenario and update the owners of vaults and queues accordingly.

Resolution

Alongside Team: Acknowledged. We can update the owner by calling directly to the vaults we want to change their owner.

I-09 | Contract Not 4626 Compliant

Category	Severity	Location	Status
Best Practices	● Info	src/UniversalVault.sol	Acknowledged

Description

Although the contracts are not intended to be fully compliant with `ERC-4626`, a few minor modifications could bring the vault closer to compliance:

1. Rename the `underlyingAsset` function to `asset()`.

2. Ensure that the max functions (`maxDeposit`, `maxMint`, `maxWithdraw`, and `maxRedeem`) return 0 when the contract is paused. Currently, deposit and withdraw operations revert as expected during a pause, meaning that users can technically deposit or withdraw 0 assets. However, the max functions return non-zero values, which creates an inconsistency.

Recommendation

Consider implementing those changes in order to make it easier for other projects to integrate.

Resolution

Alongside Team: We acknowledge this, we consider it will be almost impossible to be 100% compliant with ERC4626 due to the async withdrawal mechanism.

L-01-R | Accumulated Price Can Exceed Interval

Category	Severity	Location	Status
Logical Error	● Low	VaultFactory.sol	Resolved

Description

Function recalculateAccumulatedPrice had the upper bound validation changed from if(_upperBound > $.nextFinalizedWithdrawalId) revert InvalidUpperBound(); to if(_upperBound > $.nextWithdrawalId) revert InvalidUpperBound();

Because the nextWithdrawalId can be much greater than the maximum id that has been requested for finalization, recalculateAccumulatedPrice will provide an inaccurate accumulation for a particular interval.

Consider the following example:

1. 5 requests to withdraw for 100e18 have been made and the withdrawal ids are as following: [0, 1, 2, 3, 4, 5]

2. Manager calls requestFinalizeWithdraw with _upToWithdrawalId = 0

3. nextFinalizationRequestId is now 1

4. Off-chain script triggers recalculateAccumulatedPrice and passes upper bound with id 4

5. Accumulated amount (assuming price of 1e18) is 100e18 * 5 rather than 100e18

6. During claims too much will be withdrawn by the user and consequent users will experience ERC20InsufficientBalance reverts.

Recommendation

Be extremely careful with the inputs from the off-chain scripts, or update the validation accordingly.

Resolution

Alongside Team: The issue was resolved in PR#19.

I-01-R | Suffix Typo

Category	Severity	Location	Status
Informational	● Info	VaultFactory.sol: 26-27	Resolved

Description

`NAME_SUFIX` and `SYMBOL_SUFIX` both have a typo in the variable names. It should be `NAME_SUFFIX` and `SYMBOL_SUFFIX` respectively.

Recommendation

Update the variable naming.

Resolution

Alongside Team: The issue was resolved in commit acd7a89.

I-02-R | Visibility Conventions

Category	Severity	Location	Status
Informational	● Info	VaultFactory.sol: 160	Resolved

Description

The underscore that prefixes the function name in the `_getBytecode` function indicates that the function will be either internal or private. However, the function is public.

Recommendation

Remove the underscore to follow the same visibility conventions used elsewhere in the contract.

Resolution

Alongside Team: The issue was resolved in commit 3ba7d0b.

I-03-R | Incorrect Natspec

Category	Severity	Location	Status
Informational	● Info	VaultOracle.sol: 273	Resolved

Description

The NatSpec comment for the registerNewVault function indicates that the function is internal. However, the function is actually external.

Recommendation

Update the comment to indicate that the function is external.

Resolution

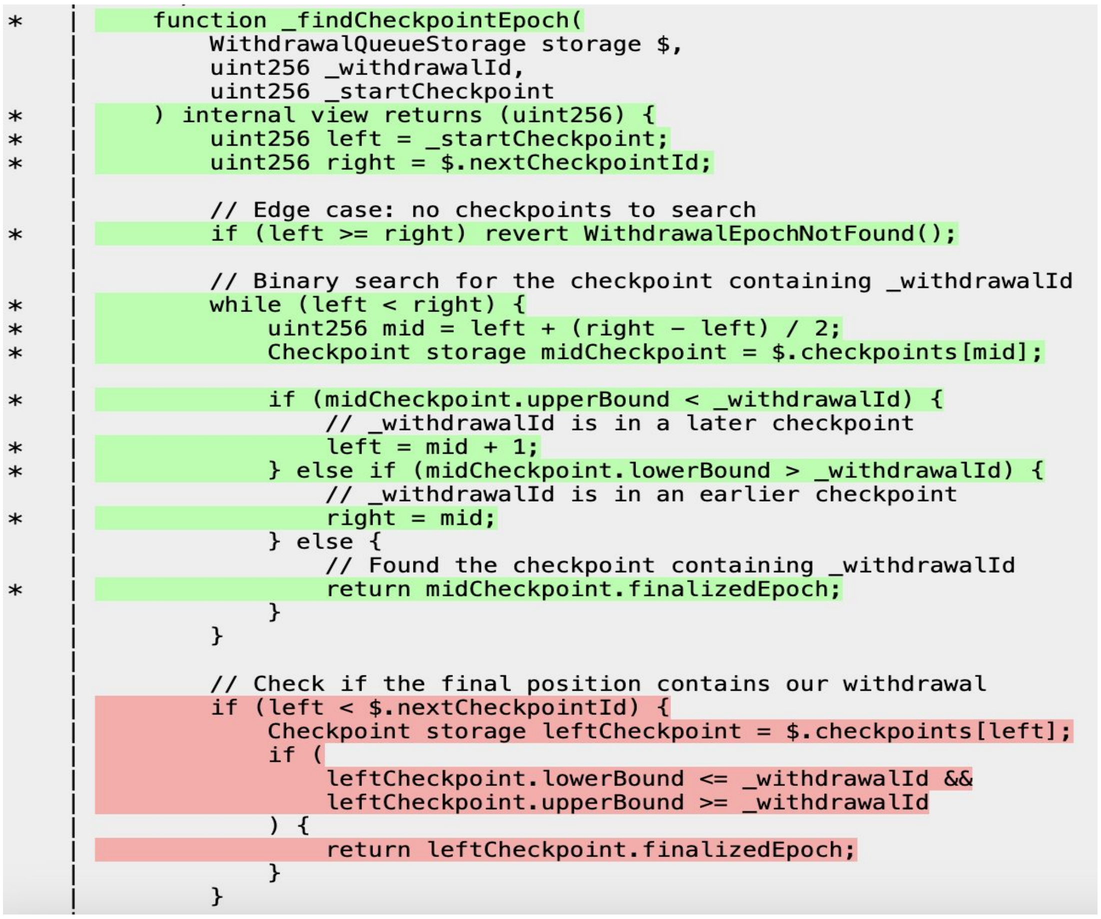
Alongside Team: The issue was resolved in commit 9ffdf75.

I-04-R | Unreachable Code

Category	Severity	Location	Status
Informational	● Info	WithdrawalQueue.sol: 604-609	Resolved

Description

The code below the binary search in `_findCheckpointEpoch` can not be hit. After many fuzzing runs, this section of code had not achieved execution.



```
*   |     function _findCheckpointEpoch(
    |         WithdrawalQueueStorage storage $,
    |         uint256 _withdrawalId,
    |         uint256 _startCheckpoint
*   |     ) internal view returns (uint256) {
*   |         uint256 left = _startCheckpoint;
*   |         uint256 right = $.nextCheckpointId;
    |
    |         // Edge case: no checkpoints to search
*   |         if (left >= right) revert WithdrawalEpochNotFound();
    |
    |         // Binary search for the checkpoint containing _withdrawalId
*   |         while (left < right) {
*   |             uint256 mid = left + (right - left) / 2;
*   |             Checkpoint storage midCheckpoint = $.checkpoints[mid];
    |
*   |             if (midCheckpoint.upperBound < _withdrawalId) {
    |                 // _withdrawalId is in a later checkpoint
*   |                 left = mid + 1;
*   |             } else if (midCheckpoint.lowerBound > _withdrawalId) {
    |                 // _withdrawalId is in an earlier checkpoint
*   |                 right = mid;
    |             } else {
    |                 // Found the checkpoint containing _withdrawalId
*   |                 return midCheckpoint.finalizedEpoch;
    |             }
    |         }
    |
    |         // Check if the final position contains our withdrawal
    |         if (left < $.nextCheckpointId) {
    |             Checkpoint storage leftCheckpoint = $.checkpoints[left];
    |             if (
    |                 leftCheckpoint.lowerBound <= _withdrawalId &&
    |                 leftCheckpoint.upperBound >= _withdrawalId
    |             ) {
    |                 return leftCheckpoint.finalizedEpoch;
    |             }
    |         }
```

Recommendation

Consider removing the code if verified to be unreachable.

Resolution

Alongside Team: The issue was resolved in commit 1b326e7.

I-05-R | No Two Vaults Can Have Same Name And Symbol

Category	Severity	Location	Status
Documentation	● Info	VaultFactory.sol: 79	Acknowledged

Description

Function deployVault deploys the Vault and WithdrawalQueue with salts based on the _name and _symbol, hence any attempted deployment with the same name and symbol and on the same chain would lead to a CREATE2 collision.

This is not an issue since only the owner can utilize the factory and existing deployments can be upgraded, but should be clearly communicated internally.

Recommendation

Be aware of this behavior.

Resolution

Alongside Team: Acknowledged.

I-06-R | Redundant Activity Check

Category	Severity	Location	Status
Documentation	● Info	VaultOracle.sol: 336	Resolved

Description

The validation `if ($.vaults[_vaultAddr].active = false) revert VaultNotActive();` was added to function `deactivateVault`, but it already has modifier `onlyActiveVault`.

Recommendation

Remove the redundant validation.

Resolution

Alongside Team: The issue was resolved in commit 8285c91.

I-07-R | Vault Pause And Activation Asymmetry

Category	Severity	Location	Status
Documentation	● Info	Global	Acknowledged

Description

When a vault is deactivated within the Vault Oracle, price updates, finalization requests, and completions are prevented.

When a vault is paused, users cannot deposit nor initiate withdrawals from the vault, cannot request, withdraw nor claim from the queue, but prices can continue to be updated and epochs can advance.

Because there is more than one way to prevent the same functionality with slight differences, it should be clearly defined when the vault is expected to be paused and when it is expected to be deactivated through the oracle.

Recommendation

Clearly document this behavior.

Resolution

Alongside Team: Acknowledged. We will document this properly in the front-end.

Disclaimer

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Guardian to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Guardian’s position is that each company and individual are responsible for their own due diligence and continuous security. Guardian’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by Guardian is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

Notice that smart contracts deployed on the blockchain are not resistant from internal/external exploit. Notice that active smart contract owner privileges constitute an elevated impact to any smart contract’s safety and security. Therefore, Guardian does not guarantee the explicit security of the audited smart contract, regardless of the verdict.

About Guardian

Founded in 2022 by DeFi experts, Guardian is a leading audit firm in the DeFi smart contract space. With every audit report, Guardian upholds best-in-class security while achieving our mission to relentlessly secure DeFi.

To learn more, visit https://guardianaudits.com

To view our audit portfolio, visit https://github.com/guardianaudits

To book an audit, message https://t.me/guardianaudits