Report

v. 2.0

Customer

Aave

# Smart Contract Audit

# Gho

2nd February 2023

Report prepared by

ABDK Consulting

# Contents

# 1 Changelog

| # | Date | Author | Description |
|---|---|---|---|
| 0.1 | 01.02.23 | A. Zveryanskaya | Initial Draft |
| 0.2 | 01.02.23 | A. Zveryanskaya | Minor revision |
| 1.0 | 01.02.23 | A. Zveryanskaya | Release |
| 1.1 | 01.02.23 | A. Zveryanskaya | Page 7. Original commit fixed |
| 1.2 | 01.02.23 | A. Zveryanskaya | Page 7. List of files is fixed |
| 1.3 | 01.02.23 | A. Zveryanskaya | Page 8. Severity levels description is added |
| 1.4 | 01.02.23 | A. Zveryanskaya | CVF-1 description fixed |
| 1.5 | 01.02.23 | A. Zveryanskaya | CVF-2, 6, 8 are downgraded |
| 2.0 | 02.02.23 | A. Zveryanskaya | Release |

# 2 Introduction



The following document provides the result of the audit performed by ABDK Consulting (Mikhail Vladimirov and Dmitry Khovratovich) at the customer request. The audit goal is a general review of the smart contracts structure, critical/major bugs detection and issuing the general recommendations.

# 3 Project scope

We were asked to review:

- Original Code
- Code with Fixes

Files:

| | | | |
|---|---|---|---|
| **facilitators/aave/stkAaveUpgrade/** | | | |
| | StakedAaveV2Rev4.sol | | |
| **facilitators/aave/tokens/** | | | |
| | GhoAToken.sol | GhoVariableDebt Token.sol | |
| **facilitators/aave/tokens/interfaces/** | | | |
| | IGhoDiscountRate Strategy.sol | IGhoAToken.sol | IGhoVariableDebt Token.sol |
| **facilitators/flashMinter/interfaces/** | | | |
| | IGhoFlashMinter.sol | | |
| **facilitators/flashMinter/** | | | |
| | GhoFlashMinter.sol | | |
| **gho/** | | | |
| | ERC20.sol | GhoToken.sol | |
| **gho/interfaces/** | | | |
| | IERC20Burnable.sol | IERC20Mintable.sol | IGhoFacilitator.sol |
| | IGhoToken.sol | | |

# 4 Methodology

The methodology is not a strict formal procedure, but rather a selection of methods and tactics combined differently and tuned for each particular project, depending on the project structure and technologies used, as well as on client expectations from the audit.

- **General Code Assessment**. The code is reviewed for clarity, consistency, style, and for whether it follows best code practices applicable to the particular programming language used. We check indentation, naming convention, commented code blocks, code duplication, confusing names, confusing, irrelevant, or missing comments etc. At this phase we also understand overall code structure.
- **Entity Usage Analysis**. Usages of various entities defined in the code are analysed. This includes both: internal usages from other parts of the code as well as potential external usages. We check that entities are defined in proper places as well as their visibility scopes and access levels are relevant. At this phase, we understand overall system architecture and how different parts of the code are related to each other.
- **Access Control Analysis**. For those entities, that could be accessed externally, access control measures are analysed. We check that access control is relevant and done properly. At this phase, we understand user roles and permissions, as well as what assets the system ought to protect.
- **Code Logic Analysis**. The code logic of particular functions is analysed for correctness and efficiency. We check if code actually does what it is supposed to do, if that algorithms are optimal and correct, and if proper data types are used. We also make sure that external libraries used in the code are up to date and relevant to the tasks they solve in the code. At this phase we also understand data structures used and the purposes they are used for.

We classify issues by the following severity levels:

- **Critical issue** directly affects the smart contract functionality and may cause a significant loss.
- **Major issue** is either a solid performance problem or a sign of misuse: a slight code modification or environment change may lead to loss of funds or data. Sometimes it is an abuse of unclear code behaviour which should be double checked.
- **Moderate issue** is not an immediate problem, but rather suboptimal performance in edge cases, an obviously bad code practice, or a situation where the code is correct only in certain business flows.
- **Minor issues** contain code style, best practices and other recommendations.

# 5 Our findings

We found 6 major issues and a few less important onesnfr. All identified major issues have been fixed or otherwise addressed in collaboration with the client.

| | Active | Fixed |
|---|---|---|
| Major | 4 | 2 |
| Moderate | 6 | 1 |
| Minor | 62 | 4 |

Fixed 7 out of 79 issues

# 6 Major Issues

## CVF-1. INFO

- **Category** Procedural
- **Source** ERC20.sol

**Description** Using compiler-enforced over-/underflow checks for checking business-evel constraints is a bad practice for the following reasons: 1. It makes the code harder to read and error-prone, as important business constraints are checked implicitly. 2. It doesn't yield a meaningful error message.

**Recommendation** Consider performing all business-level checks explicitly and then performing calculations inside an "unchecked" block.

**Client Comment** *The ERC20 code prioritizes efficiency in this case, due to most of its functions being user-facing and it is likely to be largely used. There is no other downside in this approach than making the code less readable. In case of a failed check, the code reverts.*

```
76  balanceOf[msg.sender] -= amount;
```

```
96  if (allowed != type(uint256).max) allowance[from][msg.sender] =
        ↪ allowed - amount;
```

```
98  balanceOf[from] -= amount;
```

```
171 totalSupply += amount;
```

```
183 balanceOf[from] -= amount;
```

## CVF-3. INFO

- **Category** Unclear behavior
- **Source** GhoVariableDebtToken.sol

**Recommendation** As zero AToken address has a special meaning, consider adding an explicit "require" statement to ensure that "ghoAToken" is not zero.

**Client Comment** *This function is designed to be called just once after initialization by the Aave DAO. Thus, we can assume the value of the passed address will be carefully reviewed.*

```
225 function setAToken(address ghoAToken) external override
        ↪ onlyPoolAdmin {
```

## CVF-4. INFO

- **Category** Suboptimal
- **Source** GhoVariableDebtToken.sol

**Description** A very similar pattern is coded three times.

**Recommendation** Consider extracting to a function.

**Client Comment** *It is preferable to keep these functions as they are so they do not become more difficult to read. Extracting to a function would require a great number of parameters to pass which could lead to confusion.*

```
281   (balanceIncrease, discountScaled) = _accrueDebtOnAction(
```

```
288   _burn(sender, discountScaled.toUint128());
```

```
290   _refreshDiscountPercent(
```

```
302   (balanceIncrease, discountScaled) = _accrueDebtOnAction(
```

```
309   _burn(recipient, discountScaled.toUint128());
```

```
311   _refreshDiscountPercent(
```

```
351 (uint256 balanceIncrease, uint256 discountScaled) =
      ↪ _accrueDebtOnAction(
```

```
358 _burn(user, discountScaled.toUint128());
```

```
360 _refreshDiscountPercent(
```

## CVF-5. FIXED

- **Category** Overflow/Underflow
- **Source** GhoVariableDebtToken.sol

**Description** Phantom overflow is possible here.

**Recommendation** Consider using the ”muldiv” function.

```
508 discountScaled = (discount * WadRayMath.RAY) / index;
```

## CVF-7. FIXED

- **Category** Unclear behavior
- **Source** IGhoToken.sol

**Description** Indexed event parameters of dynamic types works weird, as only the hashes of there values are actually included into an event.

**Recommendation** Consider either making this parameter non-indexed, or explicitly replacing it with a hash.

```
27 string indexed label,
```

## CVF-86. INFO

- **Category** Unclear behavior
- **Source** IERC20Mintable.sol

**Description** Minting token for an address other than the caller means that the caller address wont appear in events.

**Recommendation** Consider minting to the caller and then transferring.

**Client Comment** *Only approved Facilitators can mint GHO tokens, and it is possible to keep track of their activity thanks to the FacilitatorBucketLevelUpdated event.*

```
10 * @param account The address to create tokens for
```

# 7 Moderate Issues

## CVF-2. FIXED

- **Category** Suboptimal
- **Source** GhoAToken.sol

**Description** This conditional operator optimizes a very rare case while making the most common case more expensive.

**Recommendation** Consider removing it.

**Client Comment** *This function can be simplified to return always 0 given that GHO cannot be supplied and aGhoTokens will never be minted.*

```
143 if (currentSupplyScaled == 0) {
```

## CVF-6. INFO

- **Category** Overflow/Underflow
- **Source** GhoVariableDebtToken.sol

**Description** Overflow is possible here.

**Recommendation** Consider using a checked conversion.

**Client Comment** *The discount lock period is designed to be short and configurable by the Aave DAO. Thus, we can assume the value of the discount lock period will not pose an issue.*

```
544 uint40 newRebalanceTimestamp = uint40(block.timestamp +
    ↪ _discountLockPeriod);
```

## CVF-8. INFO

- **Category** Suboptimal
- **Source** IGhoToken.sol

**Description** This event misses the ”bucketLevel” parameter.

**Recommendation** Consider adding it.

**Client Comment** *Facilitators are added with an initial level of 0 because they are not able to mint GHO before the addition to the system. Thus, adding the level parameter to the event would be unnecessary.*

```
25 event FacilitatorAdded(
```

## CVF-9. INFO

- **Category** Unclear behavior
- **Source** GhoToken.sol

**Description** The event logs the newbucketlevel value that has not been taken mod 2^128, potentially being inconsistent with the storage.

**Client Comment** *The bucket’s capacity of the facilitator is chosen by the Aave DAO. Given that the capacity is the upper bound of the level, it will never go beyond the 2**128 value.*

```
41 emit FacilitatorBucketLevelUpdated(msg.sender, currentBucketLevel,
   ↪ newBucketLevel);
```

```
56 emit FacilitatorBucketLevelUpdated(msg.sender, currentBucketLevel,
   ↪ newBucketLevel);
```

## CVF-10. INFO

- **Category** Suboptimal
- **Source** GhoAToken.sol

**Description** The returned value is ignored here.

**Recommendation** Consider explicitly checking that the returned value is zero.

**Client Comment** *GHO token reverts if fails, so there is no need to check the returned value.*

```
189 IERC20(_underlyingAsset).transfer(_ghoTreasury, balance);
```

## CVF-11. INFO

- **Category** Suboptimal
- **Source** GhoFlashMinter.sol

**Description** The returned value is ignored.

**Recommendation** Consider explicitly checking that true is returned.

**Client Comment** *GHO token reverts if fails, so there is no need to check the returned value.*

```
101 GHO_TOKEN.transferFrom(address(receiver), address(this), amount +
        ↪ fee);
```

```
118 GHO_TOKEN.transfer(_ghoTreasury, balance);
```

## CVF-12. INFO

- **Category** Unclear behavior
- **Source** GhoVariableDebtToken.sol

**Description** In case ”sender” and ”recipient” are the same, this value could be inaccurate.

**Recommendation** Consider handling this case properly or explicitly forbidding it.

**Client Comment** *Adding a condition would optimize a very rare case while making the most common case more expensive.*

```
304 recipientPreviousScaledBalance,
```

# 8 Minor Issues

## CVF-13. FIXED

- **Category** Documentation
- **Source** GhoToken.sol

**Recommendation** It is a good practice to put a comment into an empty block to explain why the block is empty.

```
23 constructor() ERC20('Gho Token', 'GHO', 18) {}
```

## CVF-14. INFO

- **Category** Overflow/Underflow
- **Source** GhoToken.sol

**Description** Overflow is possible here.

**Recommendation** Consider asserting that the amount is not too big.

```
40 f.bucketLevel = uint128(newBucketLevel);
```

## CVF-15. INFO

- **Category** Procedural
- **Source** GhoToken.sol

**Description** Here a compile-enforced underflow check is used to check a business-level constraint.

**Recommendation** Consider using an explicit require statement instead, as the current approach makes the code harder to read and more error-prone. It also throws rather than revert in case of a failed check.

```
58 uint256 newBucketLevel = currentBucketLevel - amount;
```

## CVF-16. INFO

- **Category** Suboptimal
- **Source** GhoToken.sol

**Description** This check makes it redundant to pass the bucked level along with a facilitator config.

**Recommendation** Consider not passing it.

```
74 require(facilitatorConfig.bucketLevel == 0, '
      ↪ INVALID_BUCKET_CONFIGURATION');
```

## CVF-17. INFO

- **Category** Suboptimal
- **Source** GhoToken.sol

**Description** This event is emitted even if nothing actually changed.

**Client Comment** *This function is designed to be called by the Aave DAO. Thus, we can assume that events are emitted only if needed.*

```
115 emit FacilitatorBucketCapacityUpdated(facilitator, oldCapacity,
      ↪ newCapacity);
```

## CVF-18. INFO

- **Category** Suboptimal
- **Source** ERC20.sol

**Recommendation** Solidity doesn't support immutable variable of dynamic types, however it is possible to store short strings in immutable variables of type bytes32: https://gist.github.com/3sGgpQ8H/567354534170905e047b299286697e19

```
17 string public name;
```

```
19 string public symbol;
```

## CVF-19. INFO

- **Category** Bad naming
- **Source** ERC20.sol

**Recommendation** Consider renaming to ”nonce” for consistency with ”balanceOf” and ”allowance”.

```
44 mapping(address => uint256) public nonces;
```

## CVF-20. INFO

- **Category** Unclear behavior
- **Source** ERC20.sol

**Description** There is no range check for this argument.

**Recommendation** Consider adding appropriate checks.

```
53 uint8 _decimals
```

## CVF-21. INFO

- **Category** Procedural
- **Source** ERC20.sol

**Description** Special treatment of the ”type(uint256).max” allowance value doesn’t comply with the ERC-20 standard.

**Recommendation** Consider removing the special treatment.

```
96 if (allowed != type(uint256).max) allowance[from][msg.sender] =
   ↪ allowed - amount;
```

## CVF-22. INFO

- **Category** Procedural
- **Source** ERC20.sol

**Recommendation** Ecrecover as well as other complex functions should not be within the unchecked block.

```
137 address recoveredAddress = ecrecover(digest, v, r, s);
```

## CVF-23. INFO

- **Category** Procedural
- **Source** ERC20.sol

**Recommendation** This line could be moved out from the unchecked block, just replace ”recoveredAddress” with ”owner”.

```
141 allowance[recoveredAddress][spender] = value;
```

## CVF-24. FIXED

- **Category** Procedural
- **Source** GhoAToken.sol

**Description** Specifying a particular compiler version makes it harder to migrate to newer versions.

**Recommendation** Consider specifying as ”^0.8.0”.

```
2 pragma solidity 0.8.10;
```

## CVF-26. INFO

- **Category** Bad datatype
- **Source** GhoAToken.sol

**Recommendation** The type of this variable should be ”IGhoToken”.

```
38 address internal _underlyingAsset;
```

## CVF-27. INFO

- **Category** Procedural
- **Source** GhoAToken.sol

**Recommendation** Functions are usually defined after constructor.

```
45 function getRevision() internal pure virtual override returns (
   ↪ uint256) {
```

## CVF-28. INFO

- **Category** Bad datatype
- **Source** GhoAToken.sol

**Recommendation** The type of this argument should be ”IGhoToken”.

```
64 address underlyingAsset,
```

## CVF-29. INFO

- **Category** Suboptimal
- **Source** GhoAToken.sol

**Description** There is no range check for this argument.

**Recommendation** Consider adding appropriate checks.

```
66 uint8 aTokenDecimals,
```

## CVF-30. INFO

- **Category** Suboptimal
- **Source** GhoAToken.sol

**Description** This argument is used only to emit an event.

**Recommendation** Consider removing this argument.

**Client Comment** *This param is used to pass custom parameters when inheriting the AToken contract.*

```
69 bytes calldata params
```

## CVF-31. INFO

- **Category** Suboptimal
- **Source** GhoAToken.sol

**Description** This check makes the ”initializingPool” argument redundant.

```
71 require(initializingPool == POOL, Errors.POOL_ADDRESSES_DO_NOT_MATCH
   ↪ );
```

## CVF-32. INFO

- **Category** Bad datatype
- **Source** GhoAToken.sol

**Recommendation** The type of this argument should be ”IERC20”.

```
260 address token,
```

## CVF-33. INFO

- **Category** Suboptimal
- **Source** GhoAToken.sol

**Recommendation** As a zero variable debt token address has a special meaning, consider explicitly requiring ”ghoVariableDebtToken” to be non zero.

**Client Comment** *This function will be called just once after initialization by the Aave governance. Thus, we can assume the value of the passed address will be carefully reviewed.*

```
269 function setVariableDebtToken(address ghoVariableDebtToken) external
    ↪  override onlyPoolAdmin {
```

## CVF-34. INFO

- **Category** Unclear behavior
- **Source** GhoAToken.sol

**Description** This event is emitted even if nothing actually changed.

**Client Comment** *This function is designed to be called by the Aave DAO. Thus, we can assume that events are emitted only if needed.*

```
284 emit GhoTreasuryUpdated(oldGhoTreasury, newGhoTreasury);
```

## CVF-36. INFO

- **Category** Bad datatype
- **Source** GhoFlashMinter.sol

**Recommendation** The type of this argument should be ”IGhoToken”.

```
61 address ghoToken,
```

## CVF-37. INFO

- **Category** Bad datatype
- **Source** GhoFlashMinter.sol

**Recommendation** The type of this argumetn should be ”PoolAddressesProvider” or an interface extracted from it.

```
64 address addressesProvider
```

## CVF-38. INFO

- **Category** Bad naming
- **Source** GhoFlashMinter.sol

**Description** The semantics of the returned value is unclear.

**Recommendation** Consider giving a descriptive name to the returned value and/or describing in the documentation comment.

**Client Comment** *The function adheres to the ERC3156 spec, where the returned value is documented*

```
90 ) external override returns (bool) {
```

## CVF-39. INFO

- **Category** Procedural
- **Source** GhoFlashMinter.sol

**Recommendation** Consider replacing ”address(GHO_TOKEN)” with ”token” for compliance with the standard.

**Client Comment** *No issue here given that the token is asserted to be GHO_TOKEN.*

```
97 receiver.onFlashLoan(msg.sender, address(GHO_TOKEN), amount, fee,
     ↪ data) == CALLBACK_SUCCESS,
```

## CVF-40. INFO

- **Category** Unclear behavior
- **Source** GhoFlashMinter.sol

**Description** These events are emitted even if nothing actually changed.

**Client Comment** *This function is designed to be called by the Aave DAO. Thus, we can assume that events are emitted only if needed.*

```
127 emit FeeUpdated(oldFee, newFee);
```

```
139 emit GhoTreasuryUpdated(oldGhoTreasury, newGhoTreasury);
```

## CVF-41. FIXED

- **Category** Procedural
- **Source** GhoVariableDebtToken.sol

**Description** Specifying a particular compiler version makes it harder to migrate to newer versions.

**Recommendation** Consider specifying as ”^0.8.0”.

```
2 pragma solidity 0.8.10;
```

## CVF-43. INFO

- **Category** Bad datatype
- **Source** GhoVariableDebtToken.sol

**Recommendation** The type of this variable should be ”IAToken”.

```
37 address internal _ghoAToken;
```

## CVF-44. INFO

- **Category** Suboptimal
- **Source** GhoVariableDebtToken.sol

**Description** This check makes the ”initializingPool” argument redundant.

```
97 require(initializingPool == POOL, Errors.POOL_ADDRESSES_DO_NOT_MATCH
     ↪ );
```

## CVF-45. INFO

- **Category** Suboptimal
- **Source** GhoVariableDebtToken.sol

**Recommendation** These two return statements could be merged into one like this: if (index != previousIndex) { ... } return balance;

```
135   return balance;
```

```
145 return balance;
```

## CVF-46. INFO

- **Category** Suboptimal
- **Source** GhoVariableDebtToken.sol

**Recommendation** This line could be simplified using the ”-=” operator.

```
142 balance = balance - discount;
```

## CVF-47. INFO

- **Category** Bad naming
- **Source** GhoVariableDebtToken.sol

**Description** The semantics of the returned values is unclear.

**Recommendation** Consider giving descriptive names to the returned values and/or describing in the documentation comment.

**Client Comment** *The returned value is documented in the interface IVariableDebtToken.*

```
154 ) external virtual override onlyPool returns (bool, uint256) {
```

## CVF-48. INFO

- **Category** Bad naming
- **Source** GhoVariableDebtToken.sol

**Description** The semantics of the returned value is unclear.

**Recommendation** Consider giving a descriptive name to the returned value and/or describing in the documentation comment.

**Client Comment** *The returned value is documented in the interface IVariableDebtToken.*

```
166 ) external virtual override onlyPool returns (uint256) {
```

## CVF-49. INFO

- **Category** Suboptimal
- **Source** GhoVariableDebtToken.sol

**Description** These events are emitted even if nothing actually changed.

**Client Comment** *This function is designed to be called by the Aave DAO. Thus, we can assume that events are emitted only if needed.*

```
243 _discountRateStrategy = IGhoDiscountRateStrategy(
      ↪ newDiscountRateStrategy);
```

```
256 emit DiscountTokenUpdated(oldDiscountToken, newDiscountToken);
```

```
375 emit DiscountLockPeriodUpdated(oldLockPeriod, newLockPeriod);
```

## CVF-50. INFO

- **Category** Suboptimal
- **Source** GhoVariableDebtToken.sol

**Description** The "super.balanceOf(sender)" value is obtained twice: one before burning token, and once after.

**Recommendation** Consider calculating the "after" value from the "before" value by subtracting the known burn amount.

```
272 uint256 senderPreviousScaledBalance = super.balanceOf(sender);
```

```
292     super.balanceOf(sender).rayMul(index),
```

## CVF-51. INFO

- **Category** Suboptimal
- **Source** GhoVariableDebtToken.sol

**Description** The "super.balanceOf(recipient)" value is obtained twice: one before burning token, and once after.

**Recommendation** Consider calculating the "after" value from the "before" value by subtracting the known burn amount.

```
273 uint256 recipientPreviousScaledBalance = super.balanceOf(recipient);
```

```
313     super.balanceOf(recipient).rayMul(index),
```

## CVF-52. INFO

- **Category** Suboptimal
- **Source** GhoVariableDebtToken.sol

**Description** The expression "_ghoUserState[user]" is calculated twice.

**Recommendation** Consider using the "-=" operator.

**Client Comment** *It is preferable to keep it as it is for the correct casting.*

```
335 _ghoUserState[user].accumulatedDebtInterest = (_ghoUserState[user].
    ↪ accumulatedDebtInterest -
```

## CVF-53. INFO

- **Category** Unclear behavior
- **Source** GhoVariableDebtToken.sol

**Recommendation** Should probably be ”<=”.

```
342 _ghoUserState[user].rebalanceTimestamp < block.timestamp,
```

## CVF-54. INFO

- **Category** Suboptimal
- **Source** GhoVariableDebtToken.sol

**Description** The value ”_ghoUserState[user].rebalanceTimestamp” is obtained twice.

**Recommendation** Consider obtaining once and reusing.

```
342   _ghoUserState[user].rebalanceTimestamp < block.timestamp,
```

```
345 require(_ghoUserState[user].rebalanceTimestamp != 0, '
    ↪ NO_USER_DISCOUNT_TO_REBALANCE');
```

## CVF-55. INFO

- **Category** Suboptimal
- **Source** GhoVariableDebtToken.sol

**Description** The ”super.balanceOf(user)” value is obtained twice: one before burning token, and once after.

**Recommendation** Consider calculating the ”after” value from the ”before” value by subtracting the known burn amount.

```
348 uint256 previousScaledBalance = super.balanceOf(user);
```

```
362   super.balanceOf(user).rayMul(index),
```

## CVF-57. INFO

- **Category** Suboptimal
- **Source** GhoVariableDebtToken.sol

**Description** The ”super.balanceOf(onBehalfOf)” value is obtained twice: one before burning token, and once after.

**Recommendation** Consider calculating the ”after” value from the ”before” value by applying the known mint/burn amount.

```
405 uint256 previousScaledBalance = super.balanceOf(onBehalfOf);
```

```
423   super.balanceOf(onBehalfOf).rayMul(index),
```

## CVF-58. INFO

- **Category** Suboptimal
- **Source** GhoVariableDebtToken.sol

**Description** The ”super.balanceOf(user)” value is obtained twice: one before burning token, and once after.

**Recommendation** Consider calculating the ”after” value from the ”before” value by subtracting the known burn amount.

```
453 uint256 previousScaledBalance = super.balanceOf(user);
```

```
466   super.balanceOf(user).rayMul(index),
```

## CVF-59. INFO

- **Category** Suboptimal
- **Source** GhoVariableDebtToken.sol

**Recommendation** This calculation could be simplified as: previousScaledBalance.rayMul(index - _userState[user].additionalData);

```
498 uint256 balanceIncrease = previousScaledBalance.rayMul(index) -
    previousScaledBalance.rayMul(_userState[user].additionalData);
```

## CVF-60. FIXED

- **Category** Procedural
- **Source** GhoVariableDebtToken.sol

**Recommendation** Brackets are redundant here.

```
508 discountScaled = (discount * WadRayMath.RAY) / index;
```

## CVF-62. INFO

- **Category** Bad naming
- **Source** IGhoFlashMinter.sol

**Recommendation** Events are usually named via nouns, such as ”Fee” or ”GhoTreasury”.

**Client Comment** *GHO uses the same convention as the Aave Protocol, where events are named using the past tense.*

```
19 event FeeUpdated(uint256 oldFee, uint256 newFee);
```

## CVF-63. INFO

- **Category** Suboptimal
- **Source** IGhoFlashMinter.sol

**Recommendation** The old value parameters are redundant, as their values could be retrieved from the previous events.

```
19 event FeeUpdated(uint256 oldFee, uint256 newFee);
```

## CVF-64. INFO

- **Category** Bad datatype
- **Source** IGhoFlashMinter.sol

**Recommendation** The type of this parameter could be more specific.

```
32 address asset,
```

## CVF-66. INFO

- **Category** Bad naming
- **Source** IGhoToken.sol

**Recommendation** Events are usually named via nouns, such as ”Facilitator” or ”FacilitatorRemoval”.

**Client Comment** *GHO uses the same convention as the Aave Protocol, where events are named using the past tense.*

```
25 event FacilitatorAdded(
```

```
35 event FacilitatorRemoved(address indexed facilitatorAddress);
```

```
43 event FacilitatorBucketCapacityUpdated(
```

```
55 event FacilitatorBucketLevelUpdated(
```

## CVF-67. INFO

- **Category** Bad datatype
- **Source** IGhoToken.sol

**Recommendation** The ”facilitatorAddress” parameters could be more specific.

```
26   address indexed facilitatorAddress,
```

```
35 event FacilitatorRemoved(address indexed facilitatorAddress);
```

```
44   address indexed facilitatorAddress,
```

```
56 address indexed facilitatorAddress
```

## CVF-68. INFO

- **Category** Suboptimal
- **Source** IGhoToken.sol

**Recommendation** The old value parameters are redundant, as their values could be derived from the previous events.

```
45 uint256 oldCapacity
```

## CVF-69. INFO

- **Category** Bad datatype
- **Source** IGhoToken.sol

**Recommendation** The type of facilitator arguments could be more specific.

```
66 function addFacilitator(address facilitatorAddress, Facilitator
   ↪ memory facilitatorConfig)

73 function removeFacilitator(address facilitatorAddress) external;

80 function setFacilitatorBucketCapacity(address facilitator, uint128
   ↪ newCapacity) external;

87 function getFacilitator(address facilitator) external view returns (
   ↪ Facilitator memory);

95 function getFacilitatorBucket(address facilitator) external view
   ↪ returns (uint256, uint256);
```

## CVF-70. INFO

- **Category** Bad datatype
- **Source** IGhoToken.sol

**Recommendation** The return type could be more specific.

```
97 function getFacilitatorsList() external view returns (address[]
   ↪ memory);
```

## CVF-71. INFO

- **Category** Bad datatype
- **Source** IGhoVariableDebtToken.sol

**Recommendation** The type of parameters could be more specific.

```
16 event ATokenSet(address indexed aToken);
```

```
24   address indexed oldDiscountRateStrategy,
     address indexed newDiscountRateStrategy
```

```
33 event DiscountTokenUpdated(address indexed oldDiscountToken, address
   ↪  indexed newDiscountToken);
```

## CVF-72. INFO

- **Category** Bad naming
- **Source** IGhoVariableDebtToken.sol

**Recommendation** Events are usually named via nouns, such as ”AToken”, ”DiscountRateStrategy”, or ”DiscountToken”.

**Client Comment** *GHO uses the same convention as the Aave Protocol, where events are named using the past tense.*

```
16 event ATokenSet(address indexed aToken);
```

```
23 event DiscountRateStrategyUpdated(
```

```
33 event DiscountTokenUpdated(address indexed oldDiscountToken, address
   ↪  indexed newDiscountToken);
```

```
40 event DiscountLockPeriodUpdated(
```

```
52 event DiscountPercentLocked(
```

## CVF-73. INFO

- **Category** Suboptimal
- **Source** IGhoVariableDebtToken.sol

**Description** These parameters are redundant as their values could be derived from the previous events.

```
24 address indexed oldDiscountRateStrategy,
```

```
41 uint256 indexed oldDiscountLockPeriod,
```

```
54 uint256 oldDiscountPercent,
```

## CVF-74. INFO

- **Category** Suboptimal
- **Source** IGhoVariableDebtToken.sol

**Recommendation** The parameter ”oldDiscountToken” is redundant as its value could be derived from the previous event.

```
33 event DiscountTokenUpdated(address indexed oldDiscountToken, address
   ↪  indexed newDiscountToken);
```

## CVF-75. INFO

- **Category** Suboptimal
- **Source** IGhoVariableDebtToken.sol

**Recommendation** The parameter ”oldDiscountLockPeriod” is redundant as its value could be derived from the previous event.

```
41 uint256 indexed oldDiscountLockPeriod,
```

## CVF-76. INFO

- **Category** Bad datatype
- **Source** IGhoVariableDebtToken.sol

**Recommendation** The argument types could be more specific.

```
63 function setAToken(address ghoAToken) external;
```

```
75 function updateDiscountRateStrategy(address newDiscountRateStrategy)
   ↪  external;
```

```
87 function updateDiscountToken(address newDiscountToken) external;
```

## CVF-77. INFO

- **Category** Bad datatype
- **Source** IGhoVariableDebtToken.sol

**Recommendation** The return type could be more specific.

```
69 function getAToken() external view returns (address);
```

```
81 function getDiscountRateStrategy() external view returns (address);
```

```
93 function getDiscountToken() external view returns (address);
```

## CVF-79. INFO

- **Category** Bad datatype
- **Source** IGhoAToken.sol

**Recommendation** The parameter types could be more specific.

```
17 event VariableDebtTokenSet(address indexed variableDebtToken);
```

## CVF-80. INFO

- **Category** Bad naming
- **Source** IGhoAToken.sol

**Recommendation** Events are usually named via nouns, such as ”VariableDebtToken” or ”GhoTreasury”.

**Client Comment** *GHO uses the same convention as the Aave Protocol, where events are named using the past tense.*

```
17 event VariableDebtTokenSet(address indexed variableDebtToken);
```

## CVF-81. INFO

- **Category** Bad datatype
- **Source** IGhoAToken.sol

**Recommendation** The argument types could be more specific.

```
23 function setVariableDebtToken(address ghoVariableDebtToken) external
   ↪ ;
```

## CVF-82. INFO

- **Category** Bad datatype
- **Source** IGhoAToken.sol

**Recommendation** The returned types could be more specific.

```
29 function getVariableDebtToken() external view returns (address);
```

## CVF-83. INFO

- **Category** Bad naming
- **Source** IGhoFacilitator.sol

**Description** Events are usually named via nouns, such as ”Fees” or ”GhoTreasury”.

**Client Comment** *GHO uses the same convention as the Aave Protocol, where events are named using the past tense.*

```
16 event FeesDistributedToTreasury(
```

```
27 event GhoTreasuryUpdated(address indexed oldGhoTreasury, address
   ↪ indexed newGhoTreasury);
```

## CVF-84. INFO

- **Category** Bad datatype
- **Source** IGhoFacilitator.sol

**Description** The type of this parameter should be more specific.

```
18 address indexed asset,
```

## CVF-85. INFO

- **Category** Suboptimal
- **Source** IGhoFacilitator.sol

**Description** The ”oldGhoTreasury” parameter is redundant as its value could be derived from the previous events.

```
27 event GhoTreasuryUpdated(address indexed oldGhoTreasury, address
   ↪ indexed newGhoTreasury);
```

ABDK
Consulting

## About us

Established in 2016, is a leading service provider in the space of blockchain development and audit. It has contributed to numerous blockchain projects, and co-authored some widely known blockchain primitives like Poseidon hash function.

The ABDK Audit Team, led by Mikhail Vladimirov and Dmitry Khovratovich, has conducted over 40 audits of blockchain projects in Solidity, Rust, Circom, C++, JavaScript, and other languages.

## Contact


**Email**

dmitry@abdkconsulting.com

**Website**

abdk.consulting

**Twitter**

twitter.com/ABDKconsulting

**LinkedIn**

linkedin.com/company/abdk-consulting