CERTIK

Bloq

Vesper Pools

Security Assessment

January 20th, 2021

By:
Camden Smallwood @ CertiK
camden.smallwood@certik.org

Sheraz Arshad @ CertiK
sheraz.arshad@certik.org

Disclaimer

CertiK reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are 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 CertiK to perform a security review.

CertiK Reports do 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.

CertiK Reports should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

CertiK Reports represent an extensive auditing 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. CertiK’s position is that each company and individual are responsible for their own due diligence and continuous security. CertiK’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.

What is a CertiK report?

- A document describing in detail an in depth analysis of a particular piece(s) of source code provided to CertiK by a Client.
- An organized collection of testing results, analysis and inferences made about the structure, implementation and overall best practices of a particular piece of source code.
- Representation that a Client of CertiK has completed a round of auditing with the intention to increase the quality of the company/product's IT infrastructure and or source code.

Overview

Project Summary

Project Name	Bloq: Vesper Pools
Description	Vesper pools smart contracts
Platform	Etherum; Solidity, Yul

Audit Summary

Delivery Date	Jan. 20, 2021
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	2
Timeline	Jan. 11, 2021 - Jan. 14, 2021

Vulnerability Summary

Total Issues	38 - *26 resolved, 12 acknowledged*
Total Critical	0
Total Major	1 - *1 resolved*
Total Medium	29 - *20 resolved, 9 acknowledged*
Total Minor	2 - *1 resolved, 1 acknowledged*
Total Informational	6 - *4 resolved, 2 acknowledged*

Executive Summary

The Bloq Vesper Pools codebase was reassessed over the course of January 11th through 14th. The code was found to be mostly well-written, but contained multiple cases where return values were not taken into consideration, such as in the case of token allowance/transfers and uniswap token swapping.

After taking these issues into account, the Bloq team made changes to the codebase and supplied a new ZIP file to the CertiK team. Upon reviewing the modifications, we have determined that all of the issues related to ERC-20 token vulnerabilities were resolved. Issues `GGA-01` through `GGA-5` were acknowledged by the Bloq team, stating their preference to make minimal changes to the forked codebase. The issues concerning returned uniswap token transfer amounts were not acknowledged.

These issues should be evaluated prior to deployment. See the Findings Overview for more information.

Files In Scope

ID: VSP **- Path:** contracts/VSP.sol
SHA-256 1: `c3693a6bd7d6a657ae716c6d8d5bf9c32bb47db417102b19a670dceaf23591af`
SHA-256 2: `dbf052392facd0581e006867cd8f9eee2143b680e9ee022b48a0d59f7399bf10`

ID: GGA **- Path:** contracts/governor/GovernorAlpha.sol
SHA-256 1: `d6cadeb94011b7a7662dec42794dbe9afc31655510d75459807b020f8d302f2f`
SHA-256 2: `135f6025479f0dd35444ac65f67ab6866b1bb6c18fa0ad238494030311d00a43`

ID: GGT **- Path:** contracts/governor/GovernanceToken.sol
SHA-256 1: `6a45ece01a41666b267f37688525199d2418517f017a3158a2ed9490a7dbea45`
SHA-256 2: `375e2b8235b1cde7ca802d98631d5e6b5f726746de289c0179091c491511b9c2`

ID: GVS **- Path:** contracts/governor/VVSP.sol
SHA-256 1: `a4c0a5611d3255c6e56b1e9c6f1c5b0ffd3aa90a1b6ecd2fec36f0853c78ae48`
SHA-256 2: `87a12fff6401ad2fd0ad14444d0f30a2ffe793ab842d5d57214c4dbf4b3f195c`

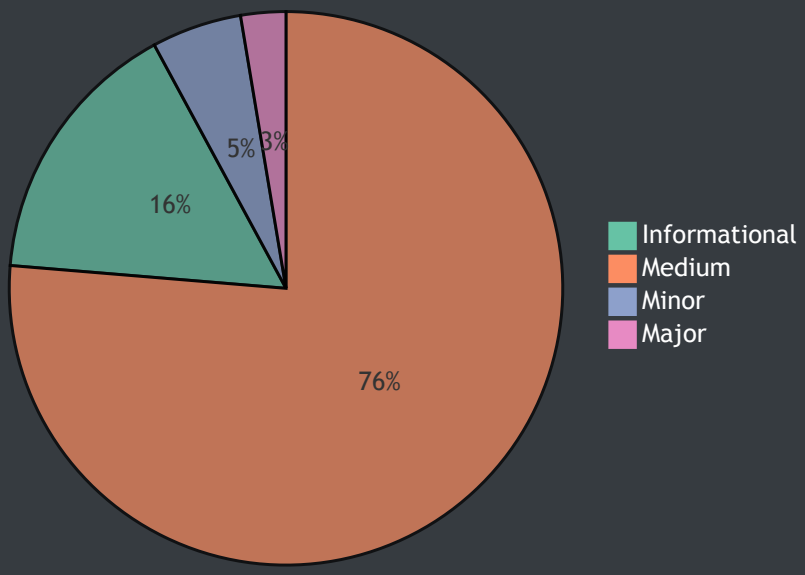
ID: AMS **- Path:** contracts/strategies/AaveMakerStrategy.sol
SHA-256 1: `e1ab9f60e2b8eadf1021d2f839b6e0e3d3e789a4eb33748ce19cdf2f15cd6496`
SHA-256 2: `571c1e7347cf9567ccbaff1409dad6c862bea0989be73f00fa7911ff8d8373bd`

ID: AMZ **- Path:** contracts/strategies/AaveV2MakerStrategy.sol
SHA-256 1: `81bf3c117bc61ba13bbd171b0fe2adaff1e818856e60b2c531f4a133dc1976e7`
SHA-256 2: `7e10adb25dd99268ed0b0f87eb6fa9801dcd196eaeea8ab0fdbf874f9fa2a0f5`

ID: AVS **- Path:** contracts/strategies/AaveV2Strategy.sol
SHA-256 1: `f836df118553ee49d7d48447b0c0a5b915c3ab93ba4edf7fd86756e39e5b904c`
SHA-256 2: `71b65378eae385001c50966e0c9756369de635c957a2ae50af45e01db92b5ec6`

ID: CMS **- Path:** contracts/strategies/CompoundStrategy.sol
SHA-256 1: `3c671cb4d8c9e1133c7fe23e66df56d97a2a97bdad86e36b992acc4efa1c0b74`
SHA-256 2: `3577a9d4f42cd68573b1dd885905524f518b89a7e32834e5dae96e5c6e686e19`

Findings



ID	Title	Type	Severity	Resolved
VSP-01	Parameter shadowing `owner` state variable	Implementation	Informational	✓
VSP-02	Non-optimal `getChainId` function implementation	Implementation	Informational	✓
GGA-01	Missing zero address validation of `guardian_` parameter in `constructor`	Volatile Code	Minor	⏱
GGA-02	Unused return value from	Volatile Code	Medium	⏱

	call to `TimelockInterface.queueTransaction`			
GGA-03	Unused return value from call to `TimelockInterface.executeTransaction`	Volatile Code	Medium	
GGA-04	Unnecessary `add256` function implementation	Implementation	Informational	
GGA-05	Unnecessary `sub256` function implementation	Implementation	Informational	
GGA-06	Non-optimal `getChainId` function implementation	Implementation	Informational	✓
GGT-01	Non-optimal `getChainId` function implementation	Implementation	Informational	✓
GVS-01	Unused return value from call to `IERC20.approve`	Volatile Code	Medium	✓
GVS-02	Unused return value from call to `IUniswapV2Router02.swapExactTokensForTokens`	Volatile Code	Medium	
GVS-03	Unused return value from call to `IERC20.approve`	Volatile Code	Medium	✓
AMS-01	Unused return value from call to `IERC20.approve`	Volatile Code	Medium	✓
AMS-02	Unused return value from call to `IERC20.approve`	Volatile Code	Medium	✓

AMS-03	Unused return value from call to `IERC20.approve`	Volatile Code	Medium	✓
AMS-04	Unused return value from call to `IERC20.transfer`	Volatile Code	Medium	✓
AMS-05	Unused return value from call to `IERC20.transfer`	Volatile Code	Medium	✓
AMS-06	Unused return value from call to `IERC20.approve`	Volatile Code	Medium	✓
AMS-07	Unused return value from call to `IUniswapV2Router02.swapExactTokensForTokens`	Volatile Code	Medium	⟳
AMS-08	Unused return value from call to `IUniswapV2Router02.swapExactTokensForTokens`	Volatile Code	Medium	⟳
AMS-09	Unused return value from call to `IERC20.transferFrom`	Volatile Code	Medium	✓
AMS-10	Unused return value from call to `IERC20.transferFrom`	Volatile Code	Medium	✓
AMZ-01	Unused return value from call to `IERC20.approve`	Volatile Code	Medium	✓
AMZ-02	Unused return value from call to `IERC20.approve`	Volatile Code	Medium	✓
AMZ-03	Unused return value from call to `IERC20.approve`	Volatile Code	Medium	✓

AMZ-04	Unused return value from call to `IERC20.transfer`	Volatile Code	Medium	✓
AMZ-05	Unused return value from call to `IERC20.approve`	Volatile Code	Medium	✓
AMZ-06	Unused return value from call to `IUniswapV2Router02.swapExactTokensForTokens`	Volatile Code	Medium	◔
AMZ-07	Unused return value from call to `IUniswapV2Router02.swapExactTokensForTokens`	Volatile Code	Medium	◔
AMZ-08	Unused return value from call to `IERC20.transferFrom`	Volatile Code	Medium	✓
AMZ-09	Unused return value from call to `IERC20.transferFrom`	Volatile Code	Medium	✓
AVS-01	Unused return value from call to `IERC20.transfer`	Volatile Code	Medium	✓
AVS-02	Unused return value from call to `IERC20.transferFrom`	Volatile Code	Medium	✓
AVS-03	Unused return value from call to `AaveLendingPool.withdraw`	Volatile Code	Medium	◔
AVS-04	No value returned from	Implementation	Major	✓

	`_updatePendingFee` function			
CMS-01	Missing zero address validation of `_rewardToken` parameter in `constructor`	Volatile Code	Minor	✓
CMS-02	Unused return value from call to `IUniswapV2Router02.swapExactTokensForTokens`	Volatile Code	Medium	↻
CMS-03	Unused return value from call to `IERC20.transfer`	Volatile Code	Medium	✓

VSP-01: Parameter shadowing `owner` state variable

Type	Severity	Location
Implementation	Informational	`contracts/VSP.sol L82`

Description:

The `permit` function in the `VSP` contract declares an `owner` address parameter, which shadows the `owner` state variable declared in the inherited `Ownable.owner` state variable. As a result, `owner` can be used incorrectly.

Recommendation:

Consider renaming the `owner` parameter in order to differentiate between the parameter supplied to the `permit` function and the `Ownable.owner` state variable:

```
    function permit(
        address _owner,
        ...
```

Alleviation:

The recommendation was found to be applied in the follow-up ZIP file supplied by the Bloq team.

VSP-02: Non-optimal `getChainId` function implementation

Type	Severity	Location
Implementation	Informational	`contracts/VSP.sol L112-L118`

Description:

The internal `getChainId` function in the `VSP` contract explicitly declares, assigns to and returns a local `uint256 chainId` variable, which is inefficient:

```
function getChainId() internal pure returns (uint256) {
    uint256 chainId;
    assembly {
        chainId := chainid()
    }
    return chainId;
}
```

Recommendation:

Consider re-declaring the `uint256 chainId` local variable as a return variable and omitting the explicit `return` statement in order to save on the overall cost of gas:

```
function getChainId() internal pure returns (uint256 chainId) {
    assembly {
        chainId := chainid()
    }
}
```

Alleviation:

The recommendation was found to be applied in the follow-up ZIP file supplied by the Bloq team.

GGA-01: Missing zero address validation of `guardian_` parameter in `constructor`

Type	Severity	Location
Volatile Code	Minor	`contracts/governor/GovernorAlpha.sol L142`

Description:

The `constructor` in the `GovernorAlpha` contract is missing zero address validation for the `guardian_` parameter, which has the potential to break the intended functionality of the `cancel`, `__acceptAdmin`, `__abdicate`, `__queueSetTimelockPendingAdmin` and `__executeSetTimelockPendingAdmin` functions.

Recommendation:

Consider adding a requirement that the supplied `guardian_` address parameter must be non-zero in order to prevent locking guardian functionality in the case of a faulty deployment:

```
    require(guardian_ != address(0));
```

Alleviation:

The issue was acknowledged by the Bloq team, who responded by informing us that this is intended functionality. The recommendation was not applied and the `GovernerAlpha` contract can still be deployed with a zero `governor` address.

GGA-02: Unused return value from call to `TimelockInterface.queueTransaction`

Type	Severity	Location
Volatile Code	Medium	`contracts/governor/GovernorAlpha.sol L257`

Description:

The internal `_queueOrRevert` function in the `VVSP` contract ignores the `bytes32` result from the call to the `TimelockInterface.queueTransaction` function on L257:

```
timelock.queueTransaction(target, value, signature, data, eta);
```

Recommendation:

Determine if the output data from queueing a proposal transaction should be taken into consideration for the `_queueOrRevert` function and incorporate it into the system if necessary, or consider returning the data from the call to the `TimelockInterface.queueTransaction` function from the `_queueOrRevert` function:

```
function _queueOrRevert(...) internal returns (bytes32) {
```

```
return timelock.queueTransaction(target, value, signature, data, eta);
```

Alleviation:

The issue was acknowledged by the Bloq team, who responded by informing us that they would prefer to apply minimal changes to the forked governor contract. As such, the recommendation was not applied and the issue still applies.

GGA-03: Unused return value from call to `TimelockInterface.executeTransaction`

Type	Severity	Location
Volatile Code	Medium	`contracts/governor/GovernorAlpha.sol L269-L275`

Description:

The external `execute` function in the `VVSP` contract ignores the `bytes memory` result from the call to the `TimelockInterface.executeTransaction` function on L269:

```
timelock.executeTransaction{value: proposal.values[i]}(
    proposal.targets[i],
    proposal.values[i],
    proposal.signatures[i],
    proposal.calldatas[i],
    proposal.eta
);
```

Recommendation:

Determine if the output data from executing each proposal transaction should be taken into consideration for the `execute` function and incorporate it into the system if necessary.

Alleviation:

The issue was acknowledged by the Bloq team, who responded by informing us that they would prefer to apply minimal changes to the forked governor contract. As such, the recommendation was not applied and the issue still applies.

GGA-04: Unnecessary `add256` function implementation

Type	Severity	Location
Implementation	Informational	`contracts/governor/GovernorAlpha.sol L444-L448`

Description:

The `GovernorAlpha` contract contains an unnecessary implementation of an `add256` function.

Recommendation:

Since the project already makes use of `@openzeppelin/contracts`, consider importing and utilizing the previously-verified OpenZeppelin `SafeMath` contract instead:

```
import "@openzeppelin/contracts/math/SafeMath.sol";
```

```
contract GovernorAlpha {
    using SafeMath for uint256;
```

Alleviation:

The issue was acknowledged by the Bloq team, who responded by informing us that they would prefer to apply minimal changes to the forked governor contract. As such, the recommendation was not applied and the issue still applies.

GGA-05: Unnecessary `sub256` function implementation

Type	Severity	Location
Implementation	Informational	`contracts/governor/GovernorAlpha.sol L450-L453`

Description:

The `GovernorAlpha` contract contains an unnecessary implementation of an `sub256` function.

Recommendation:

Since the project already makes use of `@openzeppelin/contracts`, consider importing and utilizing the previously-verified OpenZeppelin `SafeMath` contract instead:

```
import "@openzeppelin/contracts/math/SafeMath.sol";
```

```
contract GovernorAlpha {
    using SafeMath for uint256;
```

Alleviation:

The issue was acknowledged by the Bloq team, who responded by informing us that they would prefer to apply minimal changes to the forked governor contract. As such, the recommendation was not applied and the issue still applies.

GGA-06: Non-optimal `getChainId` function implementation

Type	Severity	Location
Implementation	Informational	`contracts/governor/GovernorAlpha.sol L455-L461`

Description:

The internal `getChainId` function in the `GovernorAlpha` contract explicitly declares, assigns to and returns a local `uint256 chainId` variable, which is inefficient:

```
function getChainId() internal pure returns (uint256) {
    uint256 chainId;
    assembly {
        chainId := chainid()
    }
    return chainId;
}
```

Recommendation:

Consider re-declaring the local `uint256 chainId` variable as a return variable and omitting the explicit `return` statement in order to save on the overall cost of gas:

```
function getChainId() internal pure returns (uint256 chainId) {
    assembly {
        chainId := chainid()
    }
}
```

Alleviation:

The recommendation was found to be applied in the follow-up ZIP file supplied by the Bloq team.

GGT-01: Non-optimal `getChainId` function implementation

Type	Severity	Location
Implementation	Informational	`contracts/governor/GovernanceToken.sol L217-L223`

Description:

The internal `getChainId` function in the `GovernanceToken` contract explicitly declares, assigns to and returns a local `uint256 chainId` variable, which is inefficient:

```
function getChainId() internal pure returns (uint256) {
    uint256 chainId;
    assembly {
        chainId := chainid()
    }
    return chainId;
}
```

Recommendation:

Consider re-declaring the `uint256 chainId` local variable as a return variable and omitting the explicit `return` statement in order to save on the overall cost of gas:

```
function getChainId() internal pure returns (uint256 chainId) {
    assembly {
        chainId := chainid()
    }
}
```

Alleviation:

The recommendation was found to be applied in the follow-up ZIP file supplied by the Bloq team.

GVS-01: Unused return value from call to `IERC20.approve`

Type	Severity	Location
Volatile Code	Medium	`contracts/governor/VVSP.sol L57`

Description:

The `approveToken` function in the `VVSP` contract ignores the `bool` result from the call to the `IERC20.approve` function on L57:

```
IERC20(pool).approve(strategy, MAX_UINT_VALUE);
```

Recommendation:

Reference the documentation for the `IERC20.approve` function: https://docs.openzeppelin.com/contracts/2.x/api/token/erc20#IERC20-approve-address-uint256-

Since the project makes use of the `@openzeppelin/contracts` node module, consider importing the `SafeERC20` library and using it for `IERC20` types within the `VVSP` contract, allowing the use of the `SafeERC20.safeApprove` function:

```
import "@openzeppelin/contracts/token/ERC20/SafeERC20.sol";
```

```
contract VVSP is GovernanceToken {
  use SafeERC20 for IERC20;
```

```
IERC20(pool).safeApprove(strategy, MAX_UINT_VALUE);
```

Alleviation:

The recommendation was found to be applied in the follow-up ZIP file supplied by the Bloq team.

GVS-02: Unused return value from call to `IUniswapV2Router02.swapExactTokensForTokens`

Type	Severity	Location
Volatile Code	Medium	`contracts/governor/VVSP.sol L104`

Description:

The `sweepErc20` function in the `VVSP` contract ignores the `uint256[] memory amounts` result from the call to the `IUniswapV2Router02.swapExactTokensForTokens` function on L104:

```
uniswapRouter.swapExactTokensForTokens(amt, 1, path, address(this), now + 30);
```

Recommendation:

Reference the documentation for the `IUniswapV2Router02.swapExactTokensForTokens` function: https://uniswap.org/docs/v2/smart-contracts/router02/#swapexacttokensfortokens

Determine if the output token amounts should be taken into consideration for the `sweepErc20` function and either incorporate them into the system, or consider returning the `uint256[] memory amounts` variable returned from the call to the `IUniswapV2Router02.swapExactTokensForTokens` function:

```
function sweepErc20(address _erc20) external returns (uint256[] memory) {
```

```
return uniswapRouter.swapExactTokensForTokens(amt, 1, path, address(this), now + 30);
```

Alleviation:

The issue was acknowledged by the Bloq team, who responded by informing us that they would prefer to make minimal modifications to the forked codebase. The recommendation was not taken into account and the issue still applies.

GVS-03: Unused return value from call to `IERC20.approve`

Type	Severity	Location
Volatile Code	Medium	`contracts/governor/VVSP.sol L140`

Description:

The private `_approve` function in the `VVSP` contract ignores the `bool` result from the call to the `IERC20.approve` function on L140:

```
IERC20(pool).approve(strategy, approvalAmount);
```

Recommendation:

Reference the documentation for the `IERC20.approve` function: https://docs.openzeppelin.com/contracts/2.x/api/token/erc20#IERC20-approve-address-uint256-

Since the project makes use of the `@openzeppelin/contracts` node module, consider importing the `SafeERC20` library and using it for `IERC20` types within the `VVSP` contract, allowing the use of the `SafeERC20.safeApprove` function:

```
import "@openzeppelin/contracts/token/ERC20/SafeERC20.sol";
```

```
contract VVSP is GovernanceToken {
  use SafeERC20 for IERC20;
```

```
IERC20(pool).safeApprove(strategy, approvalAmount);
```

Alleviation:

The recommendation was found to be applied in the follow-up ZIP file supplied by the Bloq team.

AMS-01: Unused return value from call to `IERC20.approve`

Type	Severity	Location
Volatile Code	Medium	`contracts/strategies/AaveMakerStrategy.sol L89`

Description:

The external `approveToken` function in the `AaveMakerStrategy` contract ignores the `bool` result from the call to the `IERC20.approve` function on L89:

```
IERC20(DAI).approve(address(cm), MAX_UINT_VALUE);
```

Recommendation:

Reference the documentation for the `IERC20.approve` function: https://docs.openzeppelin.com/contracts/2.x/api/token/erc20#IERC20-approve-address-uint256-

Since the OpenZeppelin `SafeERC20` library is already imported and used in the `AaveMakerStrategy` contract, consider utilizing the `SafeERC20.safeApprove` function in order to protect against failing approvals that do not cause a revert:

```
IERC20(DAI).safeApprove(address(cm), MAX_UINT_VALUE);
```

Alleviation:

The recommendation was found to be applied in the follow-up ZIP file supplied by the Bloq team.

AMS-02: Unused return value from call to `IERC20.approve`

Type	Severity	Location
Volatile Code	Medium	`contracts/strategies/AaveMakerStrategy.sol L95`

Description:

The external `resetApproval` function in the `AaveMakerStrategy` contract ignores the `bool` result from the call to the `IERC20.approve` function on L95:

```
IERC20(DAI).approve(address(cm), 0);
```

Recommendation:

Reference the documentation for the `IERC20.approve` function: https://docs.openzeppelin.com/contracts/2.x/api/token/erc20#IERC20-approve-address-uint256-

Since the OpenZeppelin `SafeERC20` library is already imported and used in the `AaveMakerStrategy` contract, consider utilizing the `SafeERC20.safeApprove` function in order to protect against failing approvals that do not cause a revert:

```
IERC20(DAI).safeApprove(address(cm), 0);
```

Alleviation:

The recommendation was found to be applied in the follow-up ZIP file supplied by the Bloq team.

AMS-03: Unused return value from call to `IERC20.approve`

Type	Severity	Location
Volatile Code	Medium	`contracts/strategies/AaveMakerStrategy.sol L269`

Description:

The internal `_depositDaiToAave` function in the `AaveMakerStrategy` contract ignores the `bool` result from the call to the `IERC20.approve` function on L269:

```
IERC20(DAI).approve(aavePoolCore, _amount);
```

Recommendation:

Reference the documentation for the `IERC20.approve` function: https://docs.openzeppelin.com/contracts/2.x/api/token/erc20#IERC20-approve-address-uint256-

Since the OpenZeppelin `SafeERC20` library is already imported and used in the `AaveMakerStrategy` contract, consider utilizing the `SafeERC20.safeApprove` function in order to protect against failing approvals that do not cause a revert:

```
IERC20(DAI).safeApprove(aavePoolCore, _amount);
```

Alleviation:

The recommendation was found to be applied in the follow-up ZIP file supplied by the Bloq team.

AMS-04: Unused return value from call to `IERC20.transfer`

Type	Severity	Location
Volatile Code	Medium	`contracts/strategies/AaveMakerStrategy.sol L271`

Description:

The internal `_depositDaiToAave` function in the `AaveMakerStrategy` contract ignores the `bool` result from the call to the `IERC20.transfer` function on L271:

```
    aToken.transfer(pool, _amount);
```

Recommendation:

Reference the documentation for the `IERC20.transfer` function: https://docs.openzeppelin.com/contracts/2.x/api/token/erc20#IERC20-transfer-address-uint256-

Since the OpenZeppelin `SafeERC20` library is already imported and used in the `AaveMakerStrategy` contract, consider utilizing the `SafeERC20.safeTransfer` function in order to protect against failing approvals that do not cause a revert:

```
    aToken.safeTransfer(pool, _amount);
```

Alleviation:

The recommendation was found to be applied in the follow-up ZIP file supplied by the Bloq team.

AMS-05: Unused return value from call to `IERC20.transfer`

Type	Severity	Location
Volatile Code	Medium	`contracts/strategies/AaveMakerStrategy.sol L283`

Description:

The internal `_handleFee` function in the `AaveMakerStrategy` contract ignores the `bool` result from the call to the `IERC20.transfer` function on L283:

```
IERC20(pool).transfer(controller.feeCollector(pool), feeInShare);
```

Recommendation:

Reference the documentation for the `IERC20.transfer` function: https://docs.openzeppelin.com/contracts/2.x/api/token/erc20#IERC20-transfer-address-uint256-

Since the OpenZeppelin `SafeERC20` library is already imported and used in the `AaveMakerStrategy` contract, consider utilizing the `SafeERC20.safeTransfer` function in order to protect against failing approvals that do not cause a revert:

```
IERC20(pool).safeTransfer(controller.feeCollector(pool), feeInShare);
```

Alleviation:

The recommendation was found to be applied in the follow-up ZIP file supplied by the Bloq team.

AMS-06: Unused return value from call to `IERC20.approve`

Type	Severity	Location
Volatile Code	Medium	`contracts/strategies/AaveMakerStrategy.sol L334`

Description:

The internal `_rebalanceEarned` function in the `AaveMakerStrategy` contract ignores the `bool` result from the call to the `IERC20.approve` function on L334:

```
IERC20(DAI).approve(address(uniswapRouter), balance);
```

Recommendation:

Reference the documentation for the `IERC20.approve` function: https://docs.openzeppelin.com/contracts/2.x/api/token/erc20#IERC20-approve-address-uint256-

Since the OpenZeppelin `SafeERC20` library is already imported and used in the `AaveMakerStrategy` contract, consider utilizing the `SafeERC20.safeApprove` function in order to protect against failing approvals that do not cause a revert:

```
IERC20(DAI).safeApprove(address(uniswapRouter), balance);
```

Alleviation:

The recommendation was found to be applied in the follow-up ZIP file supplied by the Bloq team.

AMS-07: Unused return value from call to `IUniswapV2Router02.swapExactTokensForTokens`

Type	Severity	Location
Volatile Code	Medium	`contracts/strategies/AaveMakerStrategy.sol L337`

Description:

The internal `_rebalanceEarned` function in the `AaveMakerStrategy` contract ignores the `uint256[] memory amounts` result from the call to the `IUniswapV2Router02.swapExactTokensForTokens` function on L337:

```
uniswapRouter.swapExactTokensForTokens(balance, 1, path, address(this),
now + 30);
```

Recommendation:

Reference the documentation for the `IUniswapV2Router02.swapExactTokensForTokens` function: https://uniswap.org/docs/v2/smart-contracts/router02/#swapexacttokensfortokens

Determine if the output token amounts should be taken into consideration for the `_rebalanceEarned` function and incorporate them into the system if necessary, such as verifying that the returned amounts match their expected values.

Alleviation:

The issue was acknowledged by the Bloq team, who responded by informing us that they would prefer to make minimal modifications to the forked codebase. The recommendation was not taken into account and the issue still applies.

AMS-08: Unused return value from call to `IUniswapV2Router02.swapExactTokensForTokens`

Type	Severity	Location
Volatile Code	Medium	`contracts/strategies/AaveMakerStrategy.sol L365`

Description:

The internal `_resurface` function in the `AaveMakerStrategy` contract ignores the `uint256[] memory amounts` result from the call to the `IUniswapV2Router02.swapExactTokensForTokens` function on L365:

```
uniswapRouter.swapExactTokensForTokens(tokenNeeded, 1, path,
address(this), now + 30);
```

Recommendation:

Reference the documentation for the `IUniswapV2Router02.swapExactTokensForTokens` function:
https://uniswap.org/docs/v2/smart-contracts/router02/#swapexacttokensfortokens

Determine if the output token amounts should be taken into consideration for the `_resurface` function and incorporate them into the system if necessary, such as verifying that the returned amounts match their expected values.

Alleviation:

The issue was acknowledged by the Bloq team, who responded by informing us that they would prefer to make minimal modifications to the forked codebase. The recommendation was not taken into account and the issue still applies.

AMS-09: Unused return value from call to `IERC20.transferFrom`

Type	Severity	Location
Volatile Code	Medium	`contracts/strategies/AaveMakerStrategy.sol L377`

Description:

The internal `_withdrawDaiFromAave` function in the `AaveMakerStrategy` contract ignores the `bool` result from the call to the `IERC20.transferFrom` function on L377:

```
    aToken.transferFrom(pool, address(this), _amount);
```

Recommendation:

Reference the documentation for the `IERC20.transfer` function: https://docs.openzeppelin.com/contracts/2.x/api/token/erc20#IERC20-transferFrom-address-address-uint256-

Since the OpenZeppelin `SafeERC20` library is already imported and used in the `AaveMakerStrategy` contract, consider utilizing the `SafeERC20.safeTransferFrom` function in order to protect against failing transfers that do not cause a revert:

```
    aToken.safeTransferFrom(pool, address(this), _amount);
```

Alleviation:

The recommendation was found to be applied in the follow-up ZIP file supplied by the Bloq team.

AMS-10: Unused return value from call to `IERC20.transferFrom`

Type	Severity	Location
Volatile Code	Medium	`contracts/strategies/AaveMakerStrategy.sol L385`

Description:

The internal `_withdrawExcessDaiFromAave` function in the `AaveMakerStrategy` contract ignores the `bool` result from the call to the `IERC20.transferFrom` function on L385:

```
    aToken.transferFrom(pool, address(this), _amount);
```

Recommendation:

Reference the documentation for the `IERC20.transfer` function: https://docs.openzeppelin.com/contracts/2.x/api/token/erc20#IERC20-transferFrom-address-address-uint256-

Since the OpenZeppelin `SafeERC20` library is already imported and used in the `AaveMakerStrategy` contract, consider utilizing the `SafeERC20.safeTransferFrom` function in order to protect against failing transfers that do not cause a revert:

```
    aToken.safeTransferFrom(pool, address(this), _amount);
```

Alleviation:

The recommendation was found to be applied in the follow-up ZIP file supplied by the Bloq team.

AMZ-01: Unused return value from call to `IERC20.approve`

Type	Severity	Location
Volatile Code	● Medium	`contracts/strategies/AaveV2MakerStrategy.sol L89`

Description:

The external `approveToken` function in the `AaveV2MakerStrategy` contract ignores the `bool` result from the call to the `IERC20.approve` function on L89:

```
IERC20(DAI).approve(address(cm), MAX_UINT_VALUE);
```

Recommendation:

Reference the documentation for the `IERC20.approve` function: https://docs.openzeppelin.com/contracts/2.x/api/token/erc20#IERC20-approve-address-uint256-

Since the OpenZeppelin `SafeERC20` library is already imported and used in the `AaveV2MakerStrategy` contract, consider utilizing the `SafeERC20.safeApprove` function in order to protect against failing approvals that do not cause a revert:

```
IERC20(DAI).safeApprove(address(cm), MAX_UINT_VALUE);
```

Alleviation:

The recommendation was found to be applied in the follow-up ZIP file supplied by the Bloq team.

AMZ-02: Unused return value from call to `IERC20.approve`

Type	Severity	Location
Volatile Code	● Medium	`contracts/strategies/AaveV2MakerStrategy.sol L95`

Description:

The external `resetApproval` function in the `AaveV2MakerStrategy` contract ignores the `bool` result from the call to the `IERC20.approve` function on L95:

```
IERC20(DAI).approve(address(cm), 0);
```

Recommendation:

Reference the documentation for the `IERC20.approve` function: https://docs.openzeppelin.com/contracts/2.x/api/token/erc20#IERC20-approve-address-uint256-

Since the OpenZeppelin `SafeERC20` library is already imported and used in the `AaveV2MakerStrategy` contract, consider utilizing the `SafeERC20.safeApprove` function in order to protect against failing approvals that do not cause a revert:

```
IERC20(DAI).safeApprove(address(cm), 0);
```

Alleviation:

The recommendation was found to be applied in the follow-up ZIP file supplied by the Bloq team.

AMZ-03: Unused return value from call to `IERC20.approve`

Type	Severity	Location
Volatile Code	● Medium	`contracts/strategies/AaveV2MakerStrategy.sol L267`

Description:

The internal `_depositDaiToAave` function in the `AaveV2MakerStrategy` contract ignores the `bool` result from the call to the `IERC20.approve` function on L267:

```
IERC20(DAI).approve(aavePoolCore, _amount);
```

Recommendation:

Reference the documentation for the `IERC20.approve` function: https://docs.openzeppelin.com/contracts/2.x/api/token/erc20#IERC20-approve-address-uint256-

Since the OpenZeppelin `SafeERC20` library is already imported and used in the `AaveV2MakerStrategy` contract, consider utilizing the `SafeERC20.safeApprove` function in order to protect against failing approvals that do not cause a revert:

```
IERC20(DAI).safeApprove(aavePoolCore, _amount);
```

Alleviation:

The recommendation was found to be applied in the follow-up ZIP file supplied by the Bloq team.

AMZ-04: Unused return value from call to `IERC20.transfer`

Type	Severity	Location
Volatile Code	● Medium	`contracts/strategies/AaveV2MakerStrategy.sol L285`

Description:

The internal `_handleFee` function in the `AaveV2MakerStrategy` contract ignores the `bool` result from the call to the `IERC20.transfer` function on L285:

```
IERC20(pool).transfer(controller.feeCollector(pool), feeInShare);
```

Recommendation:

Reference the documentation for the `IERC20.transfer` function: https://docs.openzeppelin.com/contracts/2.x/api/token/erc20#IERC20-transfer-address-uint256-

Since the OpenZeppelin `SafeERC20` library is already imported and used in the `AaveV2MakerStrategy` contract, consider utilizing the `SafeERC20.safeTransfer` function in order to protect against failing approvals that do not cause a revert:

```
IERC20(pool).safeTransfer(controller.feeCollector(pool), feeInShare);
```

Alleviation:

The recommendation was found to be applied in the follow-up ZIP file supplied by the Bloq team.

AMZ-05: Unused return value from call to `IERC20.approve`

Type	Severity	Location
Volatile Code	● Medium	`contracts/strategies/AaveV2MakerStrategy.sol L336`

Description:

The internal `_rebalanceEarned` function in the `AaveV2MakerStrategy` contract ignores the `bool` result from the call to the `IERC20.approve` function on L336:

```
IERC20(DAI).approve(address(uniswapRouter), balance);
```

Recommendation:

Reference the documentation for the `IERC20.approve` function: https://docs.openzeppelin.com/contracts/2.x/api/token/erc20#IERC20-approve-address-uint256-

Since the OpenZeppelin `SafeERC20` library is already imported and used in the `AaveV2MakerStrategy` contract, consider utilizing the `SafeERC20.safeApprove` function in order to protect against failing approvals that do not cause a revert:

```
IERC20(DAI).safeApprove(address(uniswapRouter), balance);
```

Alleviation:

The recommendation was found to be applied in the follow-up ZIP file supplied by the Bloq team.

AMZ-06: Unused return value from call to `IUniswapV2Router02.swapExactTokensForTokens`

Type	Severity	Location
Volatile Code	● Medium	`contracts/strategies/AaveV2MakerStrategy.sol L339`

Description:

The internal `_rebalanceEarned` function in the `AaveV2MakerStrategy` contract ignores the `uint256[] memory amounts` result from the call to the `IUniswapV2Router02.swapExactTokensForTokens` function on L339:

```
uniswapRouter.swapExactTokensForTokens(balance, 1, path, address(this),
now + 30);
```

Recommendation:

Reference the documentation for the `IUniswapV2Router02.swapExactTokensForTokens` function: https://uniswap.org/docs/v2/smart-contracts/router02/#swapexacttokensfortokens

Determine if the output token amounts should be taken into consideration for the `_rebalanceEarned` function and incorporate them into the system if necessary, such as verifying that the returned amounts match their expected values.

Alleviation:

The issue was acknowledged by the Bloq team, who responded by informing us that they would prefer to make minimal modifications to the forked codebase. The recommendation was not taken into account and the issue still applies.

AMZ-07: Unused return value from call to `IUniswapV2Router02.swapExactTokensForTokens`

Type	Severity	Location
Volatile Code	● Medium	`contracts/strategies/AaveV2MakerStrategy.sol L367`

Description:

The internal `_resurface` function in the `AaveV2MakerStrategy` contract ignores the `uint256[] memory amounts` result from the call to the `IUniswapV2Router02.swapExactTokensForTokens` function on L367:

```
uniswapRouter.swapExactTokensForTokens(tokenNeeded, 1, path,
address(this), now + 30);
```

Recommendation:

Reference the documentation for the `IUniswapV2Router02.swapExactTokensForTokens` function:
https://uniswap.org/docs/v2/smart-contracts/router02/#swapexacttokensfortokens

Determine if the output token amounts should be taken into consideration for the `_resurface` function and incorporate them into the system if necessary, such as verifying that the returned amounts match their expected values.

Alleviation:

The issue was acknowledged by the Bloq team, who responded by informing us that they would prefer to make minimal modifications to the forked codebase. The recommendation was not taken into account and the issue still applies.

AMZ-08: Unused return value from call to `IERC20.transferFrom`

Type	Severity	Location
Volatile Code	● Medium	`contracts/strategies/AaveV2MakerStrategy.sol L379`

Description:

The internal `_withdrawDaiFromAave` function in the `AaveV2MakerStrategy` contract ignores the `bool` result from the call to the `IERC20.transferFrom` function on L379:

```
    aToken.transferFrom(pool, address(this), _amount);
```

Recommendation:

Reference the documentation for the `IERC20.transfer` function: https://docs.openzeppelin.com/contracts/2.x/api/token/erc20#IERC20-transferFrom-address-address-uint256-

Since the OpenZeppelin `SafeERC20` library is already imported and used in the `AaveV2MakerStrategy` contract, consider utilizing the `SafeERC20.safeTransferFrom` function in order to protect against failing transfers that do not cause a revert:

```
    aToken.safeTransferFrom(pool, address(this), _amount);
```

Alleviation:

The recommendation was found to be applied in the follow-up ZIP file supplied by the Bloq team.

AMZ-09: Unused return value from call to `IERC20.transferFrom`

Type	Severity	Location
Volatile Code	● Medium	`contracts/strategies/AaveV2MakerStrategy.sol L388`

Description:

The internal `_withdrawExcessDaiFromAave` function in the `AaveV2MakerStrategy` contract ignores the `bool` result from the call to the `IERC20.transferFrom` function on L388:

```
    aToken.transferFrom(pool, address(this), _amount);
```

Recommendation:

Reference the documentation for the `IERC20.transfer` function: https://docs.openzeppelin.com/contracts/2.x/api/token/erc20#IERC20-transferFrom-address-address-uint256-

Since the OpenZeppelin `SafeERC20` library is already imported and used in the `AaveV2MakerStrategy` contract, consider utilizing the `SafeERC20.safeTransferFrom` function in order to protect against failing transfers that do not cause a revert:

```
    aToken.safeTransferFrom(pool, address(this), _amount);
```

Alleviation:

The recommendation was found to be applied in the follow-up ZIP file supplied by the Bloq team.

AVS-01: Unused return value from call to `IERC20.transfer`

Type	Severity	Location
Volatile Code	Medium	`contracts/strategies/AaveV2Strategy.sol L192`

Description:

The internal `_rebalanceEarned` function in the `AaveV2Strategy` contract ignores the `bool` result from the call to the `IERC20.transfer` function on L192:

```
IERC20(pool).transfer(controller.feeCollector(pool), feeInShare);
```

Recommendation:

Reference the documentation for the `IERC20.transfer` function: https://docs.openzeppelin.com/contracts/2.x/api/token/erc20#IERC20-transfer-address-uint256-

Since the OpenZeppelin `SafeERC20` library is already imported and used in the `AaveV2Strategy` contract, consider utilizing the `SafeERC20.safeTransfer` function in order to protect against failing transfers that do not cause a revert:

```
IERC20(pool).safeTransfer(controller.feeCollector(pool), feeInShare);
```

Alleviation:

The recommendation was found to be applied in the follow-up ZIP file supplied by the Bloq team.

AVS-02: Unused return value from call to `IERC20.transferFrom`

Type	Severity	Location
Volatile Code	Medium	`contracts/strategies/AaveV2Strategy.sol L202`

Description:

The internal `_withdraw` function in the `AaveV2Strategy` contract ignores the `bool` result from the call to the `IERC20.transferFrom` function on L202:

```
aToken.transferFrom(pool, address(this), _amount);
```

Recommendation:

Reference the documentation for the `IERC20.transfer` function: https://docs.openzeppelin.com/contracts/2.x/api/token/erc20#IERC20-transferFrom-address-address-uint256-

Since the OpenZeppelin `SafeERC20` library is already imported and used in the `AaveV2Strategy` contract, consider utilizing the `SafeERC20.safeTransferFrom` function in order to protect against failing transfers that do not cause a revert:

```
aToken.safeTransferFrom(pool, address(this), _amount);
```

Alleviation:

The recommendation was found to be applied in the follow-up ZIP file supplied by the Bloq team.

AVS-03: Unused return value from call to `AaveLendingPool.withdraw`

Type	Severity	Location
Volatile Code	Medium	`contracts/strategies/AaveV2Strategy.sol L204`

Description:

The internal `_withdraw` function in the `AaveV2Strategy` contract ignores the `uint256` result from the call to the `AaveLendingPool.withdraw` function on L204:

```
AaveLendingPool(aavePool).withdraw(address(collateralToken), _amount,
_to);
```

Recommendation:

Determine if the `uint256` value returned from the call to the `AaveLendingPool.withdraw` function should be taken into consideration for the `_withdraw` function and incorporate it into the system if necessary.

Alleviation:

The issue was acknowledged by the Bloq team, who responded by informing us that they would prefer to make minimal modifications to the forked codebase. The recommendation was not taken into account and the issue still applies.

AVS-04: No value returned from `_updatePendingFee` function

Type	Severity	Location
Implementation	Major	`contracts/strategies/AaveV2Strategy.sol L225-L227`

Description:

The internal `_updatePendingFee` function in the `AaveV2Strategy` contract specifies that it should return a `uint256` value, but does not make use of a named return variable or an explicit return statement:

```
function _updatePendingFee() internal returns (uint256) {
    pendingFee = _calculatePendingFee(aToken.balanceOf(pool));
}
```

Recommendation:

Determine if a `uint256` value is necessary to be returned from the `_updatePendingFee` function and return the appropriate value, or consider removing the return value from the function signature if it is unnecessary:

```
function _updatePendingFee() internal returns (uint256) {
    pendingFee = _calculatePendingFee(aToken.balanceOf(pool));
    return pendingFee;
}
```

```
function _updatePendingFee() internal {
    pendingFee = _calculatePendingFee(aToken.balanceOf(pool));
}
```

Alleviation:

The secondary recommendation was found to be applied in the follow-up ZIP file supplied by the Bloq team.

CERTIK CMS-01: Missing zero address validation of `_rewardToken` parameter in `constructor`

Type	Severity	Location
Volatile Code	Minor	`contracts/strategies/CompoundStrategy.sol L36`

Description:

The `constructor` in the `CompoundStrategy` contract is missing zero address validation for the `_rewardToken` parameter, which has the potential to break the intended functionality of the `sweepErc20`, `isReservedToken` and `_claimComp` functions.

Recommendation:

Consider adding a requirement that the supplied `_rewardToken` address parameter must be non-zero in order to prevent breaking reward functionality in the case of a faulty deployment:

```
    require(guardian_ != address(0));
```

Alleviation:

The recommendation was found to be applied in the follow-up ZIP file supplied by the Bloq team.

CMS-02: Unused return value from call to `IUniswapV2Router02.swapExactTokensForTokens`

Type	Severity	Location
Volatile Code	Medium	`contracts/strategies/CompoundStrategy.sol L190`

Description:

The internal `_claimComp` function in the `CompoundStrategy` contract ignores the `uint256[] memory amounts` result from the call to the `IUniswapV2Router02.swapExactTokensForTokens` function on L190:

```
uniswapRouter.swapExactTokensForTokens(amt, 1, path, address(this), now + 30);
```

Recommendation:

Reference the documentation for the `IUniswapV2Router02.swapExactTokensForTokens` function: https://uniswap.org/docs/v2/smart-contracts/router02/#swapexacttokensfortokens

Determine if the output token amounts should be taken into consideration for the `_claimComp` function and incorporate them into the system if necessary, such as verifying that the returned amounts match their expected values.

Alleviation:

The issue was acknowledged by the Bloq team, who responded by informing us that they would prefer to make minimal modifications to the forked codebase. The recommendation was not taken into account and the issue still applies.

CMS-03: Unused return value from call to `IERC20.transfer`

Type	Severity	Location
Volatile Code	Medium	`contracts/strategies/CompoundStrategy.sol L232`

Description:

The internal `_rebalanceEarned` function in the `CompoundStrategy` contract ignores the `bool` result from the call to the `IERC20.transfer` function on L232:

```
IERC20(pool).transfer(controller.feeCollector(pool), _feeInShare);
```

Recommendation:

Reference the documentation for the `IERC20.transfer` function: https://docs.openzeppelin.com/contracts/2.x/api/token/erc20#IERC20-transfer-address-uint256-

Since the OpenZeppelin `SafeERC20` library is already imported and used in the `CompoundStrategy` contract, consider utilizing the `SafeERC20.safeTransfer` function in order to protect against failing transfers that do not cause a revert:

```
IERC20(pool).safeTransfer(controller.feeCollector(pool), _feeInShare);
```

Alleviation:

The recommendation was found to be applied in the follow-up ZIP file supplied by the Bloq team.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Arithmetic

Arithmetic exhibits entail findings that relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings are exhibits that detail a fault in the logic of the linked code, such as an incorrect notion on how `block.timestamp` works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a `struct` assignment operation affecting an in-memory `struct` rather than an in-storage one.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of `private` or `delete`.

Coding Style

Coding Style findings usually do not affect the generated byte-code and comment on how to make the codebase more legible and as a result easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a `constructor` assignment imposing different `require` statements on the input variables than a setter function.

Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as `constant` contract variables aiding in their legibility and maintainability.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

Dead Code

Code that otherwise does not affect the functionality of the codebase and can be safely omitted.