[Bloq]

Security Assessment

November 11th, 2020

[Final Report]

For :

[Bloq] @ [Bloq]

By :

[Adrian Hetman] @ CertiK

adrian.hetman@certik.org

[Alex Papageorgiou] @ CertiK

alex.papageorgiou@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 indeed 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
Description	
Platform	Ethereum; Solidity, Yul
Codebase	GitHub Repository
Commits	1. b7d0729a3aeb1f71da98da854e3efba5ec64aec3 2. ebc42b289bb4912fb3ef2280b81bea60bc82ad16

Audit Summary

Delivery Date	Nov. 11th, 2020
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	2
Timeline	Oct. 27th, 2020 - Oct. 31 2020

Vulnerability Summary

Total Issues	8
Total Critical	0
Total Major	2
Total Minor	4
Total Informational	2

Findings

ID	Title	Type	Severity	Resolved
BLQ-01	Re-used interface	Implementation	Minor	✓
BLQ-02	Incorrect version of solidity	Implementation	Minor	!
BLQ-03	Lack of usage of SafeERC20 from OpenZeppelin	Implementation	Minor	! ✓
BLQ-04	Incorrect ERC20 interface	Major	Major	✓
BLQ-05	Lack of natspec comments	Implementation	Minor	✓
BLQ-06	Uses a dangerous strict equality on balance	Implementation	Major	✓
BLQ-07	Code re-use	Implementation	Informational	✓
BLQ-08	Typos in comments	Implementation	Informational	✓

BLQ-01: Re-used interface

Type	Severity	Location
Implementation	Major	AaveStrategy.sol#L15-19, AaveMakerStrategy.sol#L34-36, Controller.sol#L12-16, Controller.sol#L18-20

Description:

Interfaces of `IVesperpool` and `IStrategy` are used in a couple of places. If a codebase has two contracts the similar names, the compilation artifacts will not contain one of the contracts with the duplicate name.

Recommendation:

Our recommendation is to rename the interface name or import the whole interface and only use needed functions.

Alleviation:

Issue resolved

BLQ-02: Incorrect version of solidity

Type	Severity	Location
Implementation	Informational	General

Description:

The linked contracts necessitate a version too recent to be trusted. Consider deploying with 0.6.11. We do not recommend using any latest version for deployment, especially if changes were made in the optimizer or the language semantic. Version 0.6.12 made changes to the optimizer that's why we do not recommend using this version.

Recommendation:

Deploy with any of the following Solidity versions:

- 0.6.8,
- 0.6.10 - 0.6.11. Use a simple pragma version that allows any of these versions. Consider using the latest version of Solidity for testing.

Alleviation:

Issue unresolved.

Client comment:
"Decided to use 0.6.12. As this finding is just informational and not impacting our current code directly"

BLQ-03: Lack of usage of SafeERC20 from OpenZeppelin

Type	Severity	Location
Implementation	Minor	AaveStrategy.sol#L128, AaveStrategy.sol#L140, AaveStrategy.sol#L151,AaveStrategy.sol#L164,AaveStrategy.sol#L158, AaveStrategyETH.sol#L22, AaveStrategyETH.sol#L25, AaveStrategyETH.sol#L29, AaveStrategyETH.sol#L34,PoolRewards.sol#L90, CollateralManager.sol#L175, CollateralManager.sol#L316, CollateralManager.sol#L318, VTokenBase.sol#L52, VTokenBase.sol#L53,VTokenBase.sol#L58-L59, VTokenBase.sol#L88, VTokenBase.sol#L99,

Description:

While the ERC-20 implementation does necessitate that the `transferFrom()` / `transfer()` function returns a `bool` variable yielding `true`, many token implementations do not return anything i.e. Tether (USDT) leading to unexpected halts in code execution.

Recommendation:

We advise that the `SafeERC20.sol` library is utilized by OpenZeppelin to ensure that the `transferFrom()` / `transfer()` function is safely invoked in all circumstances.

Alleviation:

Issue partially resolved. There are stil instances where `transfer` is used instead of `safeTransfer`

Affected lines in commit hash ebc42b289bb4912fb3ef2280b81bea60bc82ad16 for:

1. Contract AaveStrategy.sol
    1. L156; L172
2. Contract PoolRewards:
    1. L90

BLQ-04: Incorrect ERC20 interface

Type	Severity	Location
Implementation	Major	IMakerDAO.sol#L6-L13, IToken.sol#L5-21

Description:

Linked contracts have an incorrect ERC20 function interface. Approve, Transfer and TransferFrom don't have return values.

Recommendation:

Add return value to the interface so it will be a proper ERC20 interface.

Alleviation:

Issue Resolved

BLQ-05: Lack of natspec comments

Type	Severity	Location
Implementation	Informational	Timelock.sol, Controller.sol, Pausable.sol, AaveStrategy.sol, PoolRewards.sol, VTokenBase.sol

Description:

Contract code is missing natspec comments, which helps understand the code and all the functions' parameters.

Recommendation:

Please follow these style guides for adding natspec comments. https://solidity.readthedocs.io/en/v0.6.11/style-guide.html?highlight=natspec#natspec

Alleviation:

Issue Resolved

BLQ-06: Uses a dangerous strict equality on balance

Type	Severity	Location
Implementation	Major	AaveStrategy.sol#L111-L113, AaveMakerStrategy.sol#L163-L165

Description:

`isEmpty()` relies on strict equality of Ether balance and total locked amount. When the owner wants to `updatePoolStrategy()`, an attacker can front-run the transaction and send ether to the contract, making `isEmpty()` return false and thus `updatePoolStrategy()` will fail. This can lead to a potential Denial of Service attack, making it impossible for the owner to update pool strategy.

```
function isEmpty() external override view returns (bool) {
        return address(this).balance == 0 && totalLocked() == 0;
    }
```

Recommendation:

Remove strict equality. It's not recommended to rely on the balance of the contract as ether always can be forcibly sent to the contract.

Alleviation:

Issue resolved.

Client comment:
"Removed eth balance check but still checking total locked to ensure that before removing strategy no collateral locked. Owner can do atomic transaction in block to avoid impact by front runner."

BLQ-07: Code re-used.

Type	Severity	Location
Implementation	Informational	CollateralManager.sol#L85, CollateralManager.sol#L114

Description:

`require(msg.sender == address(controller), "Not a controller");` is used in multiple places and could be extrated and used in a modifier.

Recommendation:

Require code that is re-used many times should be put into the own modifier.

Alleviation:

Issue resolved.

BLQ-08: Typos in comments

Type	Severity	Location
Implementation	Informational	PoolShareToken.sol#L95, PoolShareToken.sol#L107, PoolShareToken.sol#L265, PoolShareToken.sol#L279, VETH.sol#L26

Description:

In natspec comments, there is a typo in the word `retuns`.

Recommendation:

Change `retuns` to `returns`

Alleviation:

Issue resolved.

Icons explanation

 : Issue resolved

 : Issue not resolved / Acknowledged. The team will be fixing the issues in the own timeframe.

 : Issue partially resolved. Not all instances of an issue was resolved.