Security Assessment

Vee

May 26th, 2021

Table of Contents

Summary

This report has been prepared for Vee smart contracts, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

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.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	Vee
Description	Vee Finance is a new crypto asset management platform that provides a new financial service for both institutional and individual investors.
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/VeeFinance/vee-protocol
Commits	c34a3e0fcb0ee3bdb4a473518b7b3f78bc1322fa

Audit Summary

Delivery Date	May 26, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Total Issues	10
Critical	0
Major	0
Medium	0
Minor	1
Informational	9
Discussion	0

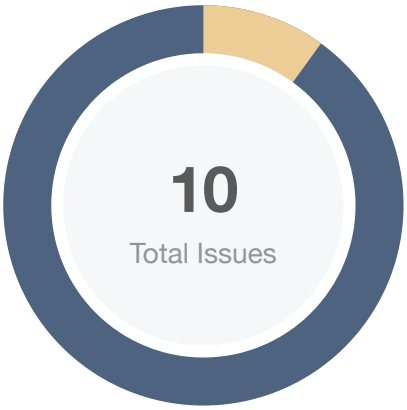
Audit Scope

ID	file	SHA256 Checksum
BJR	compound-protocol/contracts/BaseJumpRateModelV2.sol	32111c1b2bcdb051fa5c2564cd2a5e0662e699472ca5373499f67dca9c71cf47
CCL	compound-protocol/contracts/CCompLikeDelegate.sol	a54dbd1b17237582cbd618df0e5d5c925b226b81a35bc4b8ba2d241c9dd7400c
CDD	compound-protocol/contracts/CDaiDelegate.sol	c98ee33d13672016db21d4d6353b45eccb5c9f77499df77c254574a0481c0c03
CEV	compound-protocol/contracts/CErc20.sol	32a3f23addbbcb337032d185c8f07aea2144649dd871615deb68ceab75c34c9b
CED	compound-protocol/contracts/CErc20Delegate.sol	9e4f5b92705c66f910bd0c38600bede344b592f1655a07e63a6ecfad45275a3b
CEF	compound-protocol/contracts/CErc20Delegator.sol	525e15dac623328c8c5cc9591be4fc7b5af85fdb96496ac3569201b63c26614b
CEI	compound-protocol/contracts/CErc20Immutable.sol	6689cb8083354cf98dcfc49d00274046a205696451ab6df13ef3c28285c39052
CEC	compound-protocol/contracts/CEther.sol	388b46a9afc5ae6182c778e2180bc3287b3e6b959fc12c4a4377952e2b517b90
CTV	compound-protocol/contracts/CToken.sol	2791ece9eb87b766a3755ec71c5bda626d13757640b5170f6d66afc32b1fdce3
CTI	compound-protocol/contracts/CTokenInterfaces.sol	608227fbade05940ea7944840250104c0066eecf92f9f8f56ad08a5db817ed46
CMV	compound-protocol/contracts/CarefulMath.sol	dcb5b6857f6455d1daf77feb84a4cd11d3fb191fbc8097315479e88308f89083
CVF	compound-protocol/contracts/Comptroller.sol	015a86a4a9a8772529b5bb85a1ee20d2b07abb5540c6990a9a28c1d4591d3983
CIV	compound-protocol/contracts/ComptrollerInterface.sol	cb5865c24fbaf27a484b2d723172eede37694a4af38ff89a5c3447e22ad26170
CSV	compound-protocol/contracts/ComptrollerStorage.sol	294aa5754872a887eff781448983b721052ffa6b381528906968fc14eb9ce5bd

ID	file	SHA256 Checksum
DAI	compound-protocol/contracts/DAIInterestRateModelV3.sol	10d7055e4392cb87b776dc99aaeda79400008a358a5521dde3585cf30515ecb2
EIP	compound-protocol/contracts/EIP20Interface.sol	bc2ecd2927c202aab91222af287c07503cb348d8a96da3d368f195648356c4b7
EIN	compound-protocol/contracts/EIP20NonStandardInterface.sol	918d5790253d16e1b5221918d040399ad3598aec848b6a9007428965fe57e058
CGV	compound-protocol/contracts/Governance/Comp.sol	874013f6c87f2b0bf0a5d81a57fdd298ec191686cb6eed4c8498f402ef3597e6
GAG	compound-protocol/contracts/Governance/GovernorAlpha.sol	8a0553ad8bd250fc18710315dee64e3425550589c6466c01c3227fd8c7b3f1d4
GBD	compound-protocol/contracts/Governance/GovernorBravoDelegate.sol	551801cd444dcecac22a6ed5951aacb78bc6f597907a330573e5abd04b34a250
GBG	compound-protocol/contracts/Governance/GovernorBravoDelegator.sol	489be8a9c67a544ed7538d1ffb5e53cd6440ef4c33ce40e1fa27d3e5f722b09c
GBI	compound-protocol/contracts/Governance/GovernorBravoInterfaces.sol	c095701d795af25ea725b1671cacfcecd690d76eb6ddfa1fd6d7de6bfffe7e81
IRM	compound-protocol/contracts/InterestRateModel.sol	8bba52751bf2ca58e1d47012d0879a69d73e49c3de841bee79e3dfb5387b2433
JRM	compound-protocol/contracts/JumpRateModel.sol	36a81d9c51869682d7428c80357b0bd5ce9c41abb5ca51015f115fe33ae3a0e1
JRV	compound-protocol/contracts/JumpRateModelV2.sol	3c0a342bcce0fca28a0b460fc6a9c51c03eb4b3258c73140a14c0da8de242130
LIR	compound-protocol/contracts/LegacyInterestRateModel.sol	b6015e1f8ac5b818796beab7c14ccfb9aaee1f04d95216dd894c84c02d667a96
LJR	compound-protocol/contracts/LegacyJumpRateModelV2.sol	99e34556232895653e5d87a456e13858e96f1856ad55ef1157c054dfd4260541

ID	file	SHA256 Checksum
CLL	compound-protocol/contracts/Lens/CompoundLens.sol	a8e67ff444d6cf748752a49e4be8a0cf5a64ce8840e93958c76f55c702b9f9de
MVF	compound-protocol/contracts/Maximillion.sol	32f9252032165bfe274fe16f0d74b3f7add6a037b7183dc964bcf01d0a5e687c
POV	compound-protocol/contracts/PriceOracle.sol	8a5a574ee7b71ab417d5065cff4759ea32ce5c15f65e6e70fcbdd9a41d19c153
RVF	compound-protocol/contracts/Reservoir.sol	b243c40d7ab525bf64435ef35dd5e283cbed0a3085ceb52205b5fa84ba94f3ab
SMV	compound-protocol/contracts/SafeMath.sol	204a19fb7a661c5bafcd5f7916254a457ca1fd9104e5708a73dd5010b11353dc
SPO	compound-protocol/contracts/SimplePriceOracle.sol	daebe63435b50a636f65496d286461820909a3bc895166c70c49f775554c465b
TVF	compound-protocol/contracts/Timelock.sol	ea4204fc8c5c72a5f4984177c209a16be5d538f1a3ee826744c901c21d27e382
UVF	compound-protocol/contracts/Unitroller.sol	a56f8cf884f0bceb918bbb078aaa5cd3ef90002323787729d70fdee6b4a1c602
WPI	compound-protocol/contracts/WhitePaperInterestRateModel.sol	b5d06e0d725b01ecb8d0b88aa89300ddc0399904d84915a311f42f96970ba997
VPC	vee-protocol/contracts/VeeProxyController.sol	044e60219a6ba049bed1ec53f3d0383798210b3f821c8882aff1d8dbabb8ee59
VSC	vee-protocol/contracts/VeeSystemController.sol	0cd9f8b172435f9e5f0366793cd931410875c1bddb2a10b4d4039efe490fdd46

Findings



Critical	**0** (0.00%)
Major	**0** (0.00%)
Medium	**0** (0.00%)
Minor	**1** (10.00%)
Informational	**9** (90.00%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
CVF-01	Boolean equality	Gas Optimization	Informational	Acknowledged
CVF-02	Misuse of a boolean constant	Coding Style	Informational	Acknowledged
CVF-03	Return value not stored	Logical Issue	Informational	Acknowledged
CVF-04	Unlocked compiler version declaration	Language Specific	Informational	Acknowledged
CVF-05	Proper usage of `public` and `external` type	Gas Optimization	Informational	Acknowledged
CVF-06	Incorrect naming convention utilization	Coding Style	Informational	Acknowledged
CVF-07	Unused variable	Gas Optimization	Informational	Acknowledged
VPC-01	Potentially excessive permissions	**Centralization / Privilege**	**Minor**	**Resolved**
VPC-02	Missing emit events	**Centralization / Privilege**	**Informational**	**Acknowledged**
VPC-03	Constant value that could be declared as a variable	Coding Style	Informational	Acknowledged

CVF-01 | Boolean equality

Category	Severity	Location	Status
Gas Optimization	Informational	compound-protocol/contracts/Comptroller.sol: 138, 1023, 1033, 1042, 1051, 1087, 1248, 1256	Acknowledged

Description

Boolean constants can be used directly and do not need to be compared to true or false.

Example:

```
138    if (marketToJoin.accountMembership[borrower] == true) {
```

Recommendation

We recommend changing it as following:

```
    if (marketToJoin.accountMembership[borrower]) {
```

Alleviation

No Alleviation

CVF-02 | Misuse of a boolean constant

Category	Severity	Location	Status
Coding Style	Informational	compound-protocol/contracts/Comptroller.sol: 258~260, 394~396, 450~452, 519~521, 582~584, 629~631	Acknowledged

Description

Boolean constants in code have only a few legitimate uses. Other uses (in complex expressions, as conditionals) indicate either an error or, most likely, the persistence of faulty code.

Examples:

```
258   if (false) {
259     maxAssets = maxAssets;
260   }
```

Recommendation

We recommend removing the ineffectual code.

Alleviation

No Alleviation

CVF-03 | Return value not stored

Category	Severity	Location	Status
Logical Issue	Informational	compound-protocol/contracts/Comptroller.sol: 925	Acknowledged

Description

The return value of an external call is not stored in a local or state variable.

```
function _supportMarket(CToken cToken) external returns (uint) {
  ......
  cToken.isCToken(); // Sanity check to make sure its really a CToken
  ......
}
```

Recommendation

Ensure that all the return values of the function calls are used.

We recommend adding `require` statement for function `isCToken`:

```
require(cToken.isCToken(), "error message");
```

Alleviation

No Alleviation

CVF-04 | Unlocked compiler version declaration

Category	Severity	Location	Status
Language Specific	Informational	compound-protocol/contracts/Comptroller.sol: 1	Acknowledged

Description

The compiler version utilized throughout the project uses the "^" prefix specifier, denoting a greater compiler version than the version used to compile the contracts.

```
1   pragma solidity ^0.5.16;
```

Recommendation

It is a general practice to instead lock the compiler at a specific version rather than allow a range of compiler versions to be utilized to avoid compiler-specific bugs and be able to identify ones more easily. We recommend locking the compiler at the lowest possible version that supports all the capabilities wished by the codebase. This will ensure that the project utilizes a compiler version that has been in use for the longest time and as such is less likely to contain yet-undiscovered bugs.

Alleviation

No Alleviation

CVF-05 | Proper usage of `public` and `external` type

Category	Severity	Location	Status
Gas Optimization	Informational	compound-protocol/contracts/Comptroller.sol: 111, 967	Acknowledged

Description

`public` functions that are never called by the contract should be declared `external`. When the inputs are arrays, `external` functions are more efficient than `public` functions.

Functions like: `enterMarkets`, `_setMarketBorrowCaps`

Recommendation

We recommend using `external` attribute for functions never called from the contract.

Alleviation

No Alleviation

CVF-06 | Incorrect naming convention utilization

Category	Severity	Location	Status
Coding Style	Informational	compound-protocol/contracts/Comptroller.sol: 68, 71, 74, 77	Acknowledged

Description

Solidity defines a naming convention that should be followed. In general, the following naming conventions should be utilized in a Solidity file.

Constants should be named with all capital letters with underscores separating the words UPPER_CASE_WITH_UNDERSCORES.

Refer to https://solidity.readthedocs.io/en/v0.5.17/style-guide.html#naming-conventions

Alleviation

No Alleviation

CVF-07 | Unused variable

Category	Severity	Location	Status
Gas Optimization	Informational	compound-protocol/contracts/Comptroller.sol: 466, 470	Acknowledged

Description

Some unused function parameters are declared . Remove or comment out the variable name.

Recommendation

We recommend removing the unused function parameters.

Alleviation

No Alleviation

VPC-01 | Potentially excessive permissions

Category	Severity	Location	Status
Centralization / Privilege	Minor	vee-protocol/contracts/VeeProxyController.sol: 354~357	Resolved

Description

Function `setRouter` is only called by the admin, and it allows the caller to change the value of `_route`. And the `_route` address is used to swap users' assets. To improve the trustworthiness of this project, any plan to change the value of `_route` should move to the execution queue of the `Timelock`, and also add an `emit event`, or make the admin Multi-sig.

```
function setRouter(address router) external onlyAdmin {
  require(router != address(0), "setRouter: invalid token");
  _router = IUniswapV2Router02(router);
}
```

Recommendation

We recommend adding an `emit event` to the `setRouter` function. And it should transfer the admin of this contract to `Timelock`, or make the admin Multi-sig. It is better to add community voting for here.

Alleviation

The Vee team heeded our advise and changed the code. The recommendation was applied in following commits `1473b2c720a16c64d583f573313d2474781bdb93`, `b0bd522bbaaa127b78bb408d8d37fb61ba7e660d`, and `c82672787573f1ce835a94aa89f066f35d1c83eb`.

VPC-02 | Missing emit events

Category	Severity	Location	Status
Centralization / Privilege	Informational	vee-protocol/contracts/VeeProxyController.sol: 321, 332, 343	Acknowledged

Description

Several sensitive actions are defined without event declarations.

Recommendation

We recommend adding events for sensitive actions, and emit them in the functions.

Alleviation

No Alleviation

VPC-03 | Constant value that could be declared as a variable

Category	Severity	Location	Status
Coding Style	Informational	vee-protocol/contracts/VeeProxyController.sol: 550, 574, 604	Acknowledged

Description

We see the `99999999` value is used many times in the `VeeProxyController` contract.

Recommendation

We recommend declaring the value as a constant variable.

Alleviation

No Alleviation

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings 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.

Logical Issue

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

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 but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

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 CertiK 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. 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.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.