Security Assessment

DAOventures Protocol

Jun 14th, 2021

Table of Contents

Summary

This report has been prepared for DAOventures Protocol 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	DAOventures Protocol
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/daoventures/dao-protocol
Commit	c5501f78eec267b529069cf874aabb32d241db2e

Audit Summary

Delivery Date	Jun 14, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

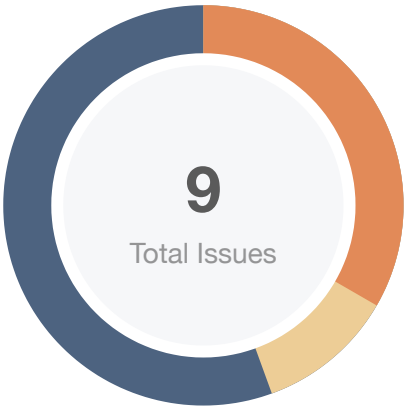
Vulnerability Summary

Total Issues	9
Critical	0
Major	3
Medium	0
Minor	1
Informational	5
Discussion	0

Audit Scope

ID	file	SHA256 Checksum
CSD	strategies/CitadelStrategy.sol	e2b4e234a77f08e06dd129b530155877a3fad7e06f04287ec8ba0b387d80a51c
CVD	vaults/CitadelVault.sol	c5851c2c85350b6e021e4ecce1067855d0f21960ad1c4b4599e7534731fd0e99

Findings



Critical	0 (0.00%)
Major	3 (33.33%)
Medium	0 (0.00%)
Minor	1 (11.11%)
Informational	5 (55.56%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
CSD-01	Flash Loans Prevention	Volatile Code	Informational	Resolved
CSD-02	Lack of Sanity Check	Volatile Code	Minor	Resolved
CSD-03	Centralized Risk	**Centralization / Privilege**	**Major**	**Acknowledged**
CSD-04	Lack of Logging Message	Coding Style	Informational	Resolved
CSD-05	Missing Emit Events	Coding Style	Informational	Resolved
CSD-06	Centralized Risk	**Centralization / Privilege**	**Major**	**Acknowledged**
CVD-01	Remaining Balance Belongs To No One	Language Specific	Informational	Resolved
CVD-02	Centralized Risk	**Centralization / Privilege**	**Major**	**Acknowledged**
CVD-03	Missing Emit Events	Coding Style	Informational	Resolved

CSD-01 | Flash Loans Prevention

Category	Severity	Location	Status
Volatile Code	Informational	strategies/CitadelStrategy.sol: 662	Resolved

Description

Flash loans are a way to borrow large amounts of money for a certain fee. The requirement is that the loans need to be returned within the same transaction in a block. If not, the transaction will be reverted.

An attacker can use the borrowed money as the initial funds for an exploit to enlarge the profit and/or manipulate the token price in the decentralized exchanges.

We find that the function `_getLPTokenPrice()` relies on price calculations that are based on-chain, meaning that they would be susceptible to flash-loan attacks by manipulating the price of given pairs to the attacker's benefit.

Recommendation

If a project requires price references, it needs to be careful of flash loans that might manipulate token prices. To prevent this from happening, we recommend the following:

1. Use Time-Weighted Average Price (TWAP). The TWAP represents the average price of a token over a specified time frame. If an attacker manipulates the price in one block, it will not affect too much on the average price.
2. If the business model allows, restrict the function caller to be a non-contract/EOA address.
3. Flash loans only allow users to borrow money within a single transaction. If the contract use cases allowed, force critical transactions to span at least two blocks.

Alleviation

`[DAOventures]`: We use the second recommendation, which restrict the public execution functions to be EOA address. Our public execution functions include `deposit()` and `withdraw()` in CitadelVault.sol. Both functions check if `msg.sender == tx.origin`.

CSD-02 | Lack of Sanity Check

Category	Severity	Location	Status
Volatile Code	Minor	strategies/CitadelStrategy.sol: 164	Resolved

Description

There's no check for the existence of communityWallet in the constructor of the contract `CitadelStrategy`.

Recommendation

We advise the client to check the existence of the communityWallet in the constructor with following snippet:

```
1 require(_communityWallet != address(0), "communityWallet does not exist");
```

Alleviation

`[DAOventures]`: The client heeded the advice and added the `require` check in the latest commit.

CSD-03 | Centralized Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	strategies/CitadelStrategy.sol: 641	**Acknowledged**

Description

The owner of the account that has the `owner` role has the privilege to call `migrateFunds()` of contract `CitadelVault.sol` to migrate all WETH to `pendingStragety`, which can be set by calling `setPendingStrategy()` after a locking period. The migration operation can be approved by `owner` role of contract `CitadelStrategy.sol` by calling function `approveMigrate()`. Any compromise to the account(s) with `owner` role of both contracts may allow the hacker to take advantage of these functions and variables, and eventually transfer all WETH to an arbitrary address.

Recommendation

We advise the client to carefully manage the account with `owner` role's private key and avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract based accounts with enhanced security practices, f.e. Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent single point of failure due to the private key;
- Introduction of a DAO / governance / voting module to increase transparency and user involvement.

Alleviation

`[DAOventures]`: The owner of CitadelVault.sol and CitadelStrategy.sol will be multi-signature wallets. Besides, we have time-lock with 48 hours latency when come to migrate funds.

CSD-04 | Lack of Logging Message

Category	Severity	Location	Status
Coding Style	Informational	strategies/CitadelStrategy.sol: 299, 301, 303	Resolved

Description

The error message in `require` checking can indicate the desired operation failure to users or relay essential warnings:

- `require(_a);`
- `require(_t);`
- `require(_s);`

Recommendation

We advise the client to provide an error message string for the `require` checking.

Alleviation

`[DAOventures]`: The client heeded the advice and added error message in `require` checks in the latest commit.

CSD-05 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	strategies/CitadelStrategy.sol: 201, 630, 636	Resolved

Description

Following functions should be able to emit events as notifications to customers because they change the status of sensitive variables. This suggestion is not limited to these codes but also applies to other similar codes.

- `setVault()`
- `setAdmin()`
- `setStrategist()`

Recommendation

We advise the client to consider adding an emit after changing the status of variables in these functions.

Alleviation

`[DAOventures]`: The client heeded the advice and added emit events in the listed functions in the latest commit.

CSD-06 | Centralized Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	strategies/CitadelStrategy.sol: 201	Acknowledged

Description

The owner of the account that has the `owner` role has the privilege to update the `vault` address by calling `setVault()` of contract `CitadelStrategy.sol` to. Any compromise to the account with the `owner` role may allow the hacker to update the sensitive value of `vault` and eventually manipulate the entire project because of the nature of the interactive relationship between the `vault` and the `strategy`

Recommendation

We advise the client to carefully manage the account with `owner` role's private key and avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract based accounts with enhanced security practices, f.e. Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent single point of failure due to the private key;
- Introduction of a DAO / governance / voting module to increase transparency and user involvement.

Alleviation

`[DAOventures]`: `setVault()` only can be execute once because of `require(address(vault) == address(0), "Vault set")`. This is to make sure the funds in CitadelStrategy won't be transfer by anyone except vault. Besides, the owner of CitadelVault.sol and CitadelStrategy.sol will be multi-signature wallets. and we have time-lock with 48 hours latency when comes to migrate funds.

CVD-01 | Remaining Balance Belongs To No One

Category	Severity	Location	Status
Language Specific	Informational	vaults/CitadelVault.sol: 227~230	Resolved

Description

When changing decimals of `_withdrawAmtInUSD` back to `1e18`, the remaining balance of user's `_withdrawAmtInUSD` will belongs to no one due to accuracy changes.

Recommendation

We would like to advise the client to notify the community that their withdrawn should be expected less than the theoretical value due to the change of value decimals.

Alleviation

`[DAOventures]`: Yes, there will be a very small amount loss due to the change of value decimals for USDT and USDC, around 0.000001 in fiat currency (USD).

CVD-02 | Centralized Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	vaults/CitadelVault.sol: 470, 443, 464	Acknowledged

Description

The owner of the account that has the `owner` role has the privilege to call `migrateFunds()` of contract `CitadelVault.sol` to migrate all WETH to `pendingStragety`, which can be set by calling `setPendingStrategy()` after a locking period. The migration operation can be approved by `owner` role of contract `CitadelStrategy.sol` by calling function `approveMigrate()`. Any compromise to the account(s) with `owner` role of both contracts may allow the hacker to take advantage of these functions and variables, and eventually transfer all WETH to an arbitrary address.

Recommendation

We advise the client to carefully manage the account with `owner` role's private key and avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract based accounts with enhanced security practices, f.e. Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent single point of failure due to the private key;
- Introduction of a DAO / governance / voting module to increase transparency and user involvement.

Alleviation

`[DAOventures]`: The owner of CitadelVault.sol and CitadelStrategy.sol will be multi-signature wallets. Besides, we have time-lock with 48 hours latency when come to migrate funds.

CVD-03 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	vaults/CitadelVault.sol: 415, 421, 427, 434, 443, 451, 457, 464	Resolved

Description

Missing Emit Events Following functions should be able to emit events as notifications to customers because they change the status of sensitive variables. This suggestion is not limited to these codes but also applies to other similar codes.

- `setTreasuryWallet()`
- `setCommunityWallet()`
- `setAdmin()`
- `setStrategist()`
- `setPendingStrategy()`
- `setBiconomy()`
- `setPercTokenKeepInVault()`
- `unlockMigrateFunds()`

Recommendation

We advise the client to consider adding an emit after changing the status of variables in these functions.

Alleviation

`[DAOventures]`: The client heeded the advice and added emit events in the listed functions in the latest commit.

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.

Volatile Code

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

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.