CERTIK

Security Assessment

DFORCE GOVERNANCE

Oct 22nd, 2021

Table of Contents

Summary

This report has been prepared for DFORCE GOVERNANCE to discover issues and vulnerabilities in the source code of the DFORCE GOVERNANCE project 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;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	DFORCE GOVERNANCE
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/dforce-network/vDFContracts
Commit	ca114a9f3727ffbdc8aff55041719f2bf18b02fc 8a9c1a68e10b69ffd3aae684a3af8944d04f51f6

Audit Summary

Delivery Date	Oct 22, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

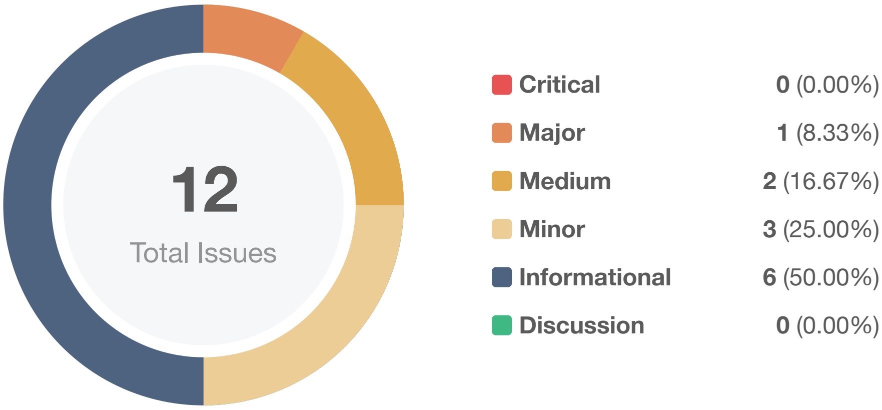
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
● Critical	0	0	0	0	0	0
● Major	1	0	0	1	0	0
● Medium	2	0	0	2	0	0
● Minor	3	0	0	3	0	0
● Informational	6	0	0	3	0	3
● Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
GTC	GovernanceToken.sol	2d44538b503785592708677cfc7e76e0bfc60da37619cc3beb1b16ca10cccb69
DFC	vDF.sol	a7216206fb7680c9e8b9e29705e1018a96f0d571b6492b14a494983c8c40a85b

Findings



ID	Title	Category	Severity	Status
GLOBAL-01	Unlocked compiler version	Language Specific	● Informational	✓ Resolved
DFC-01	Centralization risk	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
DFC-02	Logic issue in `unstakeUnderlying()`	Logical Issue	● Medium	ⓘ Acknowledged
DFC-03	Source of reward token	Logical Issue	● Medium	ⓘ Acknowledged
DFC-04	Third party dependencies	Volatile Code	● Minor	ⓘ Acknowledged
DFC-05	Incompatibility with deflationary tokens	Volatile Code	● Minor	ⓘ Acknowledged
DFC-06	Payers and receivers are different	Volatile Code	● Minor	ⓘ Acknowledged
DFC-07	Lack of reasonable boundary	Volatile Code	● Informational	✓ Resolved
DFC-08	Lack of zero address validation	Volatile Code	● Informational	✓ Resolved
GTC-01	Time check relies on timestamp	Logical Issue	● Informational	ⓘ Acknowledged
GTC-02	Declaration naming convention	Coding Style	● Informational	ⓘ Acknowledged
GTC-03	Proper usage of “public” and “external” type	Gas Optimization	● Informational	ⓘ Acknowledged

GLOBAL-01 | Unlocked compiler version

Category	Severity	Location	Status
Language Specific	● Informational	Global	✓ Resolved

Description

The contract has unlocked compiler version. An unlocked compiler version in the source code of the contract permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers. This can lead to an ambiguity when debugging as compiler specific bugs may occur in the codebase that would be hard to identify over a span of multiple compiler versions rather than a specific one.

Recommendation

We advise that the compiler version is instead locked at the lowest version possible that the contract can be compiled at. For example, for version `v0.5.16` the contract should contain the following line:

```
pragma solidity 0.5.16;
```

Alleviation

The team heeded our advice and resolved this issue in commit `8a9c1a68e10b69ffd3aae684a3af8944d04f51f6`.

DFC-01 | Centralization risk

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	vDF.sol: 89, 110	ⓘ Acknowledged

Description

In the contract `vDF`, the role `owner` has the authority over the following function:

- `setRewardRate()`: change the reward rate to affect the reward amount,
- `setNewVault()`: change the `rewardVault` address where the reward tokens come from,

Any compromise to the `owner` account may allow the hacker to take advantage of this and users' assets may suffer loss.

Recommendation

We advise the client to carefully manage the `owner` account's private key to 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 smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at the different levels in terms of short-term and long-term:

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

Alleviation

The team acknowledged this issue and they stated the following: "They are in the process of migrating to a DAO, and this is part of their DAO governance system."

DFC-02 | Logic issue in `unstakeUnderlying()`

Category	Severity	Location	Status
Logical Issue	● Medium	vDF.sol: 243~258	ⓘ Acknowledged

Description

In the below code in the function `unstakeUnderlying()` of the contract `vDF`, it calculates the amount of `vDF` token to burn for getting the exact amount(`_underlyingAmount`) of token `DF`. According to the white paper, vDF = DF / ExchangeRate, the amount of `vDF` token can be directly computed by using the `rdiv()` function like in the `stake()` function. However, the code snippet uses `rdivup()` function and add extra `_exchangeRate - 1` to the amount `_underlyingAmount`. We would like to confirm with the client if the current implementation aligns with the original project design.

```
        uint256 _exchangeRate = getCurrentExchangeRate();
        uint256 _tokenAmount = _underlyingAmount.rdivup(_exchangeRate);
```

Besides, in the function `rdivup()`, the sub-part `b.sub(1)` may be incorrect. Since the value of `b` passed by `unstakeUnderlying()` already includes the decimal value `10**18`, the `b.sub(1)` part is almost equal to `b`.

```
    function rdivup(uint256 a, uint256 b) internal pure returns (uint256 c) {
        c = a.mul(BASE).add(b.sub(1)).div(b);
    }
```

Recommendation

We recommend stating for this and fixing the logic.

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

DFC-03 | Source of reward token

Category	Severity	Location	Status
Logical Issue	● Medium	vDF.sol	ⓘ Acknowledged

Description

In the contract `vDF`, the reward tokens are the token `DF` and the reward tokens are stored in the `rewardVault` contract. However, these two contract addresses are only initialized with uncertain addresses and the owner of current contract can even modify the `rewardVault` address. When the `DF` balance in the contract `rewardVault` is insufficient, the users may not get full amount of rewards and this staking platform would seem like a zero-sum game.

Recommendation

We recommend ensuring the `DF` token balance of contract `rewardVault` is enough for reward distribution.

Also ensure that the token `DF` and contract `rewardVault` are under control of current protocol, the passed addresses are correct, and the contract implementations can meet the requirements.

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

DFC-04 | Third party dependencies

Category	Severity	Location	Status
Volatile Code	● Minor	vDF.sol: 18	ⓘ Acknowledged

Description

The contracts `vDF` are serving as the underlying entity to interact with third party the `rewardVault` contract. The scope of the audit treats 3rd party entities as black boxes and assumes their functional correctness. However, in the real world, 3rd parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of 3rd parties can possibly create severe impacts, such as increasing fees of 3rd parties, migrating to new LP pools, etc.

Recommendation

We understand that the business logic of strategy requires interaction with the `rewardVault`, etc. We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

DFC-05 | Incompatibility with deflationary tokens

Category	Severity	Location	Status
Volatile Code	● Minor	vDF.sol	ⓘ Acknowledged

Description

When transferring standard ERC20 deflationary tokens, the input amount may not be equal to the received amount due to the charged transaction fee. For example, if a user stakes 100 deflationary tokens (with a 10% transaction fee) in the `vDF` contract, only 90 tokens actually arrived in the contract. However, the user can still withdraw 100 `vDF` tokens from the contract, which causes the contract to lose 10 `vDF` tokens in such a transaction.

Recommendation

We advise the client to regulate the `DF` token and ensure that it is not a deflationary token.

Alleviation

The team acknowledged this issue and they stated the following: "The DF token is not a deflationary token."

DFC-06 | Payers and receivers are different

Category	Severity	Location	Status
Volatile Code	● Minor	vDF.sol: 191, 223, 243	ⓘ Acknowledged

Description

In the function `stake()`, the `DF` payer address is `msg.sender`, while the `vDF` receiver is `_recipient`. Also in the function `unstake()` and `unstakeUnderlying()`, the `vDF` payer address is `_from`, while the `DF` receiver is `msg.sender`. We would like to confirm with the client if the current implementation aligns with the original project design.

Recommendation

We recommend stating for this.

Alleviation

The team acknowledged this issue and they stated the following: "This is their design to be integrated with other contracts."

DFC-07 | Lack of reasonable boundary

Category	Severity	Location	Status
Volatile Code	● Informational	vDF.sol: 20, 59, 93	✓ Resolved

Description

State variable `rewardRate` can be set in the `constructor()` function and changed in the `setRewardRate` function. It will directly affect the reward amount. Thus, it would be better to have a reasonable upper and lower boundaries.

Recommendation

We recommend adding reasonable upper and lower boundaries to state variable `rewardRate`.

Alleviation

The team heeded our advice and resolved this issue in commit `8a9c1a68e10b69ffd3aae684a3af8944d04f51f6`.

DFC-08 | Lack of zero address validation

Category	Severity	Location	Status
Volatile Code	● Informational	vDF.sol: 56	✓ Resolved

Description

The input variable `_rewardVault` in `vDF.initialize()` should not be zero address, but does not have zero address validation.

Recommendation

We recommend adding zero address validation to the variable `_rewardVault`.

Alleviation

The team heeded our advice and resolved this issue in commit `8a9c1a68e10b69ffd3aae684a3af8944d04f51f6`.

GTC-01 | Time check relies on timestamp

Category	Severity	Location	Status
Logical Issue	● Informational	GovernanceToken.sol: 156	ⓘ Acknowledged

Description

Any comparison should avoid using `now` or any other timestamp as any powerful miner can dominate the mining and thus manipulate the timestamp, which will eventually lead to vulnerability as deny delegation in certain situations.

Recommendation

We recommend using block number and avoiding relying on `now` or any type of timestamp.

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

GTC-02 | Declaration naming convention

Category	Severity	Location	Status
Coding Style	● Informational	GovernanceToken.sol: 6, 9, 12	ⓘ Acknowledged

Description

The linked declarations (`name`, `symbol`, `decimals`) do not conform to the Solidity style guide with regards to its naming convention.

Particularly:

- `UPPER_CASE`: Should be applied to `constant` variables

Recommendation

We advise that the linked variable and function names are adjusted to properly conform to Solidity's naming convention.

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

GTC-03 | Proper usage of “public” and “external” type

Category	Severity	Location	Status
Gas Optimization	● Informational	GovernanceToken.sol: 136, 149, 177	ⓘ Acknowledged

Description

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

For example,

- `delegate()`,
- `delegateBySig()`,
- `getPriorVotes()`.

Recommendation

We advise the client to use the external attribute for functions never called from the contract.

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

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.

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 you ("Customer" or 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, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

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.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS

AVAILABLE” AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER’S OR ANY OTHER PERSON’S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER’S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK’S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER’S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED “AS IS” AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK’S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING

MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

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.

CERTIK