dForce Network

Lending Smart Contracts

Security Assessment

February 20th, 2021

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

Sheraz Arshad @ CertiK
sheraz.arshad@certik.org

Angelos Apostolidis @ CertiK
angelos.apostolidis@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	dForce Network - Lending Smart Contracts
Description	Smart contracts of the dForceLending repository. The code implements a lending protocol with the functionality of providing flash loans. All of the assets lent and borrowed acrrue interests based on the set the parameters.
Platform	Ethereum; Solidity, Yul
Codebase	GitHub Repository
Commits	1. d78d9b2fa1325bf3001f821ae9941bf2b5851655 2. 397ec65b5676bd2f64d72e532e961595ab931e3d

Audit Summary

Delivery Date	Feb. 20, 2021
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	3
Timeline	Jan. 2, 2021 - Jan. 31, 2021

Vulnerability Summary

Total Issues	54
Total Medium	1
Total Minor	8
Total Informational	45

Executive Summary

All of the functions in the `Controller` contract have proper access restriction and parameter sanitization where necessary. The equity was found to be calculated correctly for each of the accounts. Most of the findings are optimizational.

The `PriceOracle` contract has proper access restriction and parameter sanitization where necessary and applies proper utilization of the `Anchors` and `Reader` architecture. Most of the findings are optimizational, while POE-12 addresses specific uses of `assert` instead of `require`, which should not be used in a production environment, as they will consume all remaining gas in the event of a failure. Protection against flash loan vulnerabilities and price manipulation was found to be implemented via the swing and anchors constraints, but there is still a centralization issue with the `PriceOracle` contract. For most occasions, it would be advisable to utilize a decentralized price oracle system, such as Chainlink.

While the `Base` contract was found to have proper access restriction and parameter sanitization, BAS-01 outlines the possibility for replay attacks, which should be addressed prior to deployment.

The `RewardDistributor` contract contains the potential for re-entrancy attacks in the `claimReward` function, as outlined in RDR-01. The contract also fails to emit an event when updating the global distribution speed in the `_setGlobalDistributionSpeed` function, as outlined in RDR-02. The `_updateReward` function does not check if the supplied `_address` is non-zero, as outlined in RDR-03. All of the other findings are optimizational.

The `TokenERC20` contract has the potential for re-entrancy attacks, as outlined in TER-01, which can be resolved with a simple inclusion of the `nonReentrant` modifier.

The `iToken` contract has proper access restriction for all the functions.

The `iETH` contract was checked for native ETH transferring and receiving in order to ensure that they are properly implemented for flash loans. In particular, ETH-02 points out the usage of `transfer` for sending ETH, while ETH-05 points out the usage of `receive` for receiving ETH. Proper access restriction was found to be implemented along with correct parameter sanitization, which is missing in the actual function implementations but they are handled in their internal counterparts.

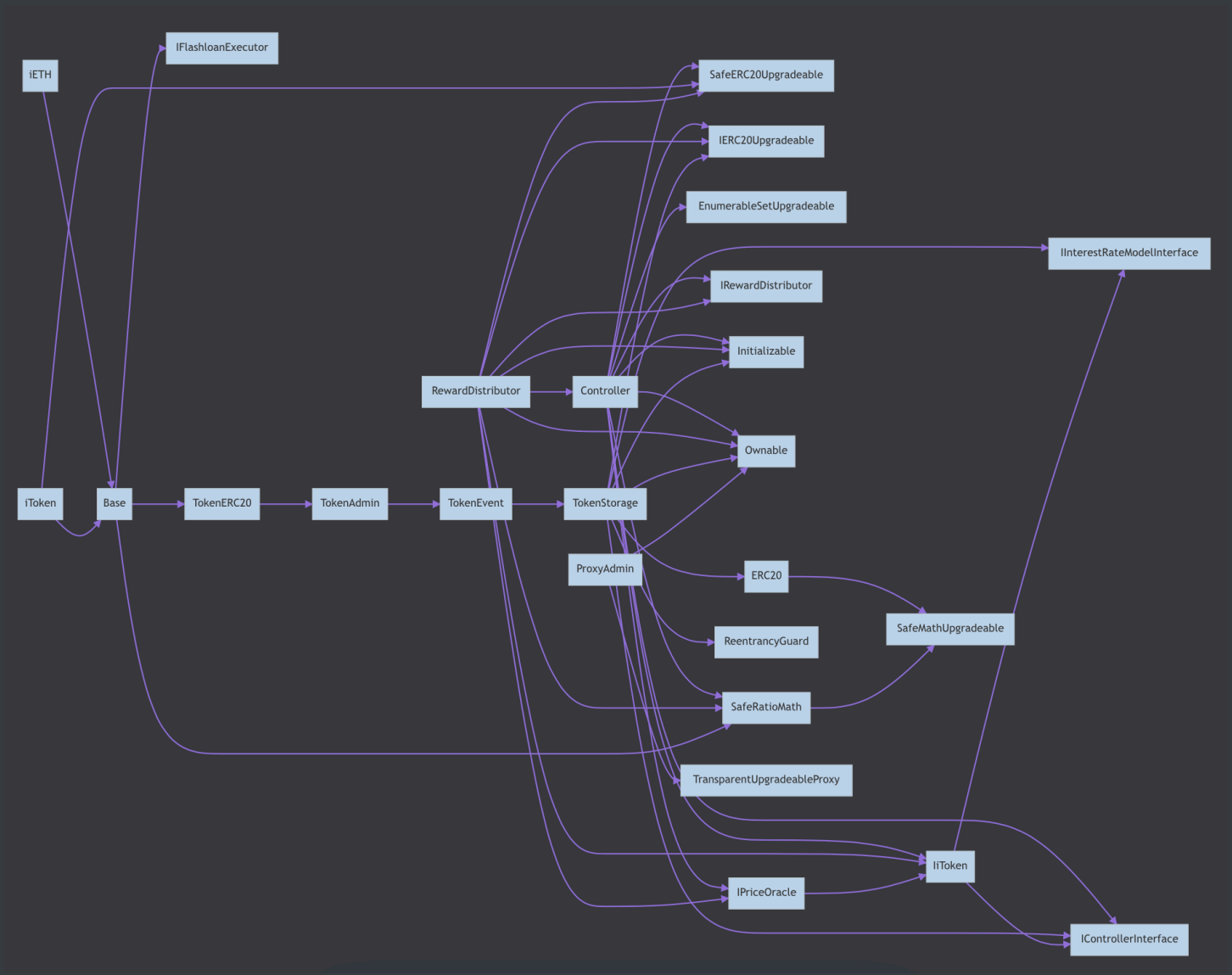
It should be noted that the `nonReentrant` modifier should be refactored on the external `liquidateBorrow` function in the `iETH` and `iToken` contracts in order to allow calling it from within the `flashloan` function. Removing the `nonReentrant` modifier from the `liquidateBorrow` function altogether is not an option, as it would open the potential for re-entrancy attacks. The `nonReentrant` modifier should be replaced with a separate `bool` state variable which acts as a mutex to prevent re-entrancy attacks within the `liquidateBorrow` function, requiring the `bool` mutex to be `false` at the beginning of `liquidateBorrow`, setting the `bool` mutex to `true` after the requirement, calling the `_liquidateBorrowInternal` function, then setting the `bool` mutex back to `false` after the call to the `_liquidateBorrowInternal` function. The reason the `nonReentrant` modifier is insufficient is because it shares a single `bool` across all functions marked `nonReentrant` within the contract, which makes it impossible to call a separate function marked `nonReentrant` when a `nonReentract` function is already executing. This is what prevents the `liquidateBorrow` function from being called from within the `flashloan` function.

A review was performed for the flashloan exploit's fix related to decreasing of `exchangeRate` enabling the liquidation of insolvent positions. The fix of adding flashloan amount to `totalBorrows` is deemed safe and does not introduce any issues in flashloan, minting, borrowing, repay and liquidation functionalitiles, and is regarded important to be incorporated for production deployment.

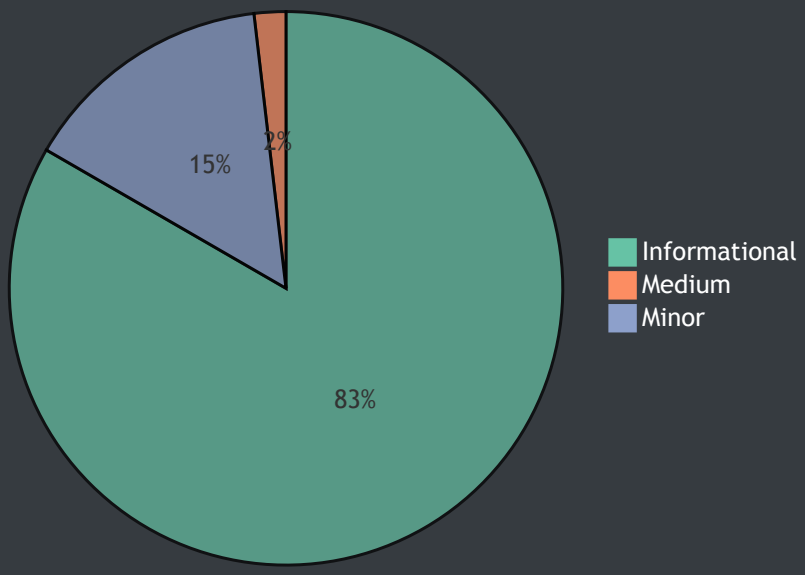
Files In Scope

ID	Contract	Location
BAS	Base.sol	contracts/TokenBase/Base.sol
CON	Controller.sol	contracts/Controller.sol
ERC	ERC20.sol	contracts/library/ERC20.sol
ITN	IiToken.sol	contracts/interface/IiToken.sol
IPO	IPriceOracle.sol	contracts/interface/IPriceOracle.sol
INI	Initializable.sol	contracts/library/Initializable.sol
IRM	InterestRateModel.sol	contracts/InterestRateModel/InterestRateModel.sol
IFE	IFlashloanExecutor.sol	contracts/interface/IFlashloanExecutor.sol
IRD	IRewardDistributor.sol	contracts/interface/IRewardDistributor.sol
ICI	IControllerInterface.sol	contracts/interface/IControllerInterface.sol
IIR	IInterestRateModelInterface.sol	contracts/interface/IInterestRateModelInterface.sol
OWN	Ownable.sol	contracts/library/Ownable.sol
PAN	ProxyAdmin.sol	contracts/library/ProxyAdmin.sol
POE	PriceOracle.sol	contracts/PriceOracle.sol
RGD	ReentrancyGuard.sol	contracts/library/ReentrancyGuard.sol
RDR	RewardDistributor.sol	contracts/RewardDistributor.sol
SRM	SafeRatioMath.sol	contracts/library/SafeRatioMath.sol
TAN	TokenAdmin.sol	contracts/TokenBase/TokenAdmin.sol
TER	TokenERC20.sol	contracts/TokenBase/TokenERC20.sol
TET	TokenEvent.sol	contracts/TokenBase/TokenEvent.sol
TSE	TokenStorage.sol	contracts/TokenBase/TokenStorage.sol
ETH	iETH.sol	contracts/iETH.sol
ITO	iToken.sol	contracts/iToken.sol

File Dependency Graph



Findings



ID	Title	Type	Severity	Resolved
CON-01	Mappings data can be packed in a struct	Gas Optimization	Informational	✓
CON-02	Mappings data can be packed in a struct	Gas Optimization	Informational	✓
CON-03	Comparison with literal `true`	Gas Optimization	Informational	✓
CON-04	Inefficient use of `require` statements	Gas Optimization	Informational	✓

CON-05	Inefficient use of `require` statements	Gas Optimization	Informational	✓
CON-06	Redundant casting to type `address`	Gas Optimization	Informational	✓
CON-07	Ineffectual code	Inconsistency	Informational	✓
CON-08	Explicitly returning local variable	Gas Optimization	Informational	✓
CON-09	Functions visiblity can be changed to `external`	Gas Optimization	Informational	✓
CON-10	Documentation discrepancy	Inconsistency	Informational	✓
CON-11	Incorrect naming convention for `external` functions	Language Specific	Informational	↻
CON-12	Lack of verification for the passed argument	Logical Issue	Minor	✓
POE-01	Code Optimization	Gas Optimization	Informational	↻
POE-02	Conditional Optimization	Gas Optimization	Informational	↻
POE-03	Visibility Specifiers Missing	Language Specific	Informational	↻
POE-04	State Layout Optimization	Gas Optimization	Informational	↻
POE-05	Order of Layout	Coding Style	Informational	↻
POE-06	Variable Visibility	Gas Optimization	Informational	↻
POE-07	`event` Optimization	Language Specific	Informational	↻
POE-08	Ambiguous Error Message	Inconsistency	Informational	↻
POE-09	Function Optimization	Gas Optimization	Informational	↻
POE-10	Code Optimization	Gas Optimization	Informational	↻

POE-11	Ambiguous NetSpec Comments	Coding Style	Informational	✓
POE-12	Introduction of `require` Statements	Volatile Code	Minor	
POE-13	Function Visibility Optimization	Gas Optimization	Informational	
POE-14	Naming Conventions	Coding Style	Informational	
POE-15	Inexistant Input Sanitization	Volatile Code	Informational	
ETH-01	Ambiguous Statement	Volatile Code	Informational	
ETH-02	Usage of `transfer()` for sending Ether	Volatile Code	Minor	
ETH-04	Non Standard Contract Naming	Coding Style	Informational	
ETH-05	Inexistent Input Sanitization	Volatile Code	Minor	✓
ETH-06	Contract Size	Compiler Error	Informational	✓
ITO-02	Redundant casting to `uint8`	Gas Optimization	Informational	✓
ITO-03	Contract name does not comply with the convention	Language Specific	Informational	
BAS-01	Possibility of replay attack in `permit`	Volatile Code	Minor	
RDR-01	Potential for re-entrancy attacks in `claimReward`	Volatile Code	Medium	✓
RDR-02	Missing event for updating global distribution speed	Implementation	Minor	✓
RDR-03	Lack of address check in	Volatile Code	Minor	✓

	`_updateReward`			
RDR-04	Unnecessary underscore prefixing `_setRewardToken`	Naming Conventions	Informational	
RDR-05	Unnecessary underscore prefixing `_addRecipient`	Naming Conventions	Informational	
RDR-06	Unnecessary underscore prefixing `_pause`	Naming Conventions	Informational	
RDR-07	Unnecessary underscore prefixing `_unpause`	Naming Conventions	Informational	
RDR-08	Unnecessary underscore prefixing `_setGlobalDistributionSpeed`	Naming Conventions	Informational	
RDR-09	Inefficient early return in `updateDistributionSpeed`	Gas Optimization	Informational	✓
RDR-10	Unnecessary underscore prefixing `_setDistributionFactors`	Naming Conventions	Informational	
RDR-13	`claimAllReward` should be declared external	Implementation	Informational	✓
IRM-01	`getBorrowRate` should be declared external	Implementation	Informational	
TAN-01	Unnecessary underscore prefixing `_setController`	Naming Conventions	Informational	
TAN-02	Unnecessary underscore	Naming Conventions	Informational	

	prefixing `_setInterestRateModel`			
TAN-03	Unnecessary underscore prefixing `_setNewReserveRatio`	Naming Conventions	Informational	
TAN-04	Unnecessary underscore prefixing `_setNewFlashloanFeeRatio`	Naming Conventions	Informational	
TAN-05	Unnecessary underscore prefixing `_setNewProtocolFeeRatio`	Naming Conventions	Informational	
TAN-06	Unnecessary underscore prefixing `_withdrawReserves`	Naming Conventions	Informational	
TER-01	Potential for re-entrancy attacks in `_transferTokens`	Volatile Code	Minor	

CON-01: mappings data can be packed in a struct

Type	Severity	Location
Gas Optimization	Informational	Controller.sol L62, L66

Description:

The mappings on the aforementioned lines have key of type `address` representing a user's address. These mappings can be combined into a single mapping having `address` as key type and the value type will be a struct having properties from both aforementioned mappings. This will reduce the lookup gas cost when reading data from these mappings.

Recommendation:

We advise to replace the aforementioned mappings with a single mapping by utilizing a struct for the value types.

```
struct UserData {
    EnumerableSetUpgradeable.AddressSet collaterals;
    EnumerableSetUpgradeable.AddressSet borrowed;
}
```

```
mapping(address => User) internal usersData;
```

Alleviation:

The recommendation was applied in commit 397ec65b5676bd2f64d72e532e961595ab931e3d.

CON-02: Mappings data can be packed in a struct

Type	Severity	Location
Gas Optimization	Informational	Controller.sol L59, L107, L110, L113

Description:

The mappings on the aforementioned lines have key of type `address` representing a market's address. These mappings can be combined into a single mapping having `address` as key type and the value type will be a struct having properties from all aforementioned mappings. This will reduce the lookup gas cost when reading data from these mappings.

Recommendation:

We advise to replace the aforementioned mappings with a single mapping by utilizing a struct for the value types.

```
struct MarketData {
    Market market;
    bool mintPaused;
    bool borrowPaused;
    bool redeemPaused;
}
```

```
mapping(address => MarketData) public marketsData;
```

Alleviation:

The recommendation was applied in commit 397ec65b5676bd2f64d72e532e961595ab931e3d.

CON-03: Comparison with literal `true`

Type	Severity	Location
Gas Optimization	Informational	Controller.sol L136

Description:

The aforementioned line performs comparison with a literal `true`. This comparison can be replaced with the expression itself to increase the legibility of the code.

Recommendation:

We advise to utilize the expression itself in place of comparison with literal `true`.

```
require(
            msg.sender == owner || _paused,
            "Only owner can unpause"
);
```

Alleviation:

The recommendation was applied in commit 397ec65b5676bd2f64d72e532e961595ab931e3d.

CON-04: Inefficient use of `require` statements

Type	Severity	Location
Gas Optimization	Informational	Controller.sol L130-L138

Description:

The `require` statements on the aforementioned lines can be replaced with a single `require` statement to increase the legibility of the codebase and optimizing deploying gas cost from reduced bytecode footprint of the contract.

Recommendation:

We advise to use a single `require` statements with the combined conditional logic from both of the aforementioned `require` statements.

```
require(
    msg.sender == owner || (msg.sender == pauseGuardian && _paused),
    "only owner can pause/unpause and only guardian can pause"
);
```

Alleviation:

The recommendation was applied in commit 397ec65b5676bd2f64d72e532e961595ab931e3d.

CON-05: Inefficient use of `require` statements

Type	Severity	Location
Gas Optimization	● Informational	Controller.sol L300, L336, L374, L395, L432, L454, L476, L588, L700, L770, L802, L1037

Description:

Th `require` statements on the aforementioned lines can substituted with a function call which would perform the said assertion. This will reduce the bytecode footprint of the contract resulting in reduced gas cost upon the deployment.

Recommendation:

We advise to introduce a private function and that be used in place of the `require` statements to reduce gas cost associated with individual use same `require` statement.

```
function _isTokenAdded(address iToken) private {
    require(iTokens.contains(_iToken), "Token has not been listed");
}
```

Alleviation:

The recommendation was applied in commit 397ec65b5676bd2f64d72e532e961595ab931e3d.

CON-06: Redundant casting to type `address`

Type	Severity	Location
Gas Optimization	Informational	Controller.sol L1067, L1072, L1079, L1410

Description:

The aforementioned lines perform redundant casting of `iToken` to type `address` which already is of type `address`.

Recommendation:

We advise to remove the redundant casting to `address` to save gas cost associated with it.

Alleviation:

The recommendation was applied in commit 397ec65b5676bd2f64d72e532e961595ab931e3d.

CON-07: Ineffectual code

Type	Severity	Location
Inconsistency	Informational	Controller.sol L601

Description:

The aforementioned line utilizes local variable `_minter` as an expression to silence the compiler warning of unused variable. As the variable is being used on `L610`, the line specifying the expression can be removed.

Recommendation:

We advise to remove the use of expression on the aforementioned line.

Alleviation:

The recommendation was applied in commit 397ec65b5676bd2f64d72e532e961595ab931e3d.

CON-08: Explicitly returning local variable

Type	Severity	Location
Gas Optimization	Informational	Controller.sol L1249, L1292, L1315, L1373, L1429, L1494

Description:

The function on the aforementioned line explicitly returns a local variable which increases the overall cost of gas.

Recommendation:

Since named return variables can be declared in the signature of a function, consider refactoring to remove the local variable declaration and explicit return statement in order to reduce the overall cost of gas.

Alleviation:

The recommendation was applied in commit 397ec65b5676bd2f64d72e532e961595ab931e3d.

CON-09: Functions visiblity can be changed to `external`

Type	Severity	Location
Gas Optimization	Informational	Controller.sol L1315, L1373

Description:

The functions on the aforementioned lines are never called within the contract and can have their visibilities changed to `external` and the data location of their array parameters can be changed to `calldata` which will save the gas cost associated with copying parameters to `memory`.

Recommendation:

We advise to change the functions' visibilites to `external` and the data location of their reference parameters to `calldata`.

Alleviation:

The recommendation was applied in commit 397ec65b5676bd2f64d72e532e961595ab931e3d.

CON-10: Documentation discrepancy

Type	Severity	Location
Inconsistency	Informational	Controller.sol L51

Description:

There is documentation discrepancy in the comment on aforementioned line which describes the property `supplyCapacity` following it as being checked in `beforeBorrow` function hook yet it is only checked in `beforeMint`.

Recommendation:

We advise to change the comment on the aforementioned line to `The supply capacity of the asset, will be checked in beforeMint()`.

Alleviation:

The recommendation was applied in commit 397ec65b5676bd2f64d72e532e961595ab931e3d.

CON-11: Incorrect naming convention for `external` functions

Type	Severity	Location
Language Specific	● Informational	Controller.sol L174, L231, L246, L267, L295, L331, L368, L389, L409, L426, L448, L470, L481, L491, L509, L527, L552

Description:

The names of `external` functions on the aforementioned lines are prefixed with underscore (_), which is a convention typically reserved for `private` and `internal` declarations.

Recommendation:

We advise to remove the `_` from ther start of the function names to comply with the naming convention for `external` functions.

Alleviation:

The recommendation was not taken into account, with the dForce team stating "Public functions starts with _ are owner functions for easy interaction on Remix or Etherscan."

CON-12: Lack of verification for the passed argument

Type	Severity	Location
Logical Issue	Minor	Controller.sol L409

Description:

The function `_setPauseGuardian` receives `_newPauseGuardian` of type address as its parameter, which is not validated against zero value.

Recommendation:

We advise to check the zero value of the argument `_newPauseGuardian` passed to the function.

```
require(
  _newPauseGuardian != address(0),
  "_newPauseGuardian cannot be zero"
);
```

Alleviation:

The recommendation was applied in commit 397ec65b5676bd2f64d72e532e961595ab931e3d.

POE-01: Code Optimization

Type	Severity	Location
Gas Optimization	Informational	PriceOracle.sol L187, L200

Description:

The `else` block is redundant, as it only contains the `return` statement that is meant to be executed in every scenario other than the one checked in the `if` block.

Recommendation:

We advise to remove the `else` block and directly use the `return` statement.

Alleviation:

The recommendation was not taken into account, with the dForce team stating "No plan to change the oracle implementation."

POE-02: Conditional Optimization

Type	Severity	Location
Gas Optimization	Informational	PriceOracle.sol L233

Description:

The linked conditional should only check against the edge case, i.e. inequality with zero.

Recommendation:

We advise to change to a "not equal" operation instead.

Alleviation:

The recommendation was not taken into account, with the dForce team stating "No plan to change the oracle implementation."

POE-03: Visibility Specifiers Missing

Type	Severity	Location
Language Specific	Informational	PriceOracle.sol L259, L262, L268, L269

Description:

The linked variable declarations do not have a visibility specifier explicitly set.

Recommendation:

Inconsistencies in the default visibility the Solidity compilers impose can cause issues in the functionality of the codebase. We advise that visibility specifiers for the linked variables are explicitly set.

Alleviation:

The recommendation was not taken into account, with the dForce team stating "No plan to change the oracle implementation."

POE-04: State Layout Optimization

Type	Severity	Location
Gas Optimization	Informational	PriceOracle.sol L495-L522

Description:

The state should be as tightly packed as possible to 256-bit sized variables.

Recommendation:

We advise to change to a more optimal state layout.

Alleviation:

The recommendation was not taken into account, with the dForce team stating "No plan to change the oracle implementation."

POE-05: Order of Layout

Type	Severity	Location
Coding Style	Informational	PriceOracle.sol General

Description:

The order of layout does not follow the Solidity conventions.

Recommendation:

We advise to closely follow the Solidity style guide.

Alleviation:

The recommendation was not taken into account, with the dForce team stating "No plan to change the oracle implementation."

POE-06: Variable Visibility

Type	Severity	Location
Gas Optimization	Informational	PriceOracle.sol L498, L505

Description:

The linked variables are only used for `internal` operations, hence can have a stricter visibility specifier to save gas.

Recommendation:

We advise to change the visibility of the linked variables to `internal`.

Alleviation:

The recommendation was not taken into account, with the dForce team stating "No plan to change the oracle implementation."

POE-07: `event` Optimization

Type	Severity	Location
Language Specific	Informational	PriceOracle.sol L602, L669, L714, L721, L731, L746, L751, L761, L776, L784, L789

Description:

The linked events could add the `address` parameters to the `topics` data structure.

Recommendation:

We advise to add the `indexed` attribute to the `address` parameters of the linked events.

Alleviation:

The recommendation was not taken into account, with the dForce team stating "No plan to change the oracle implementation."

POE-08: Ambiguous Error Message

Type	Severity	Location
Inconsistency	Informational	PriceOracle.sol L946

Description:

The error message of the linked `require` statement does not point to the problem at hand.

Recommendation:

We advise to update the linked error message.

Alleviation:

The recommendation was not taken into account, with the dForce team stating "No plan to change the oracle implementation."

POE-09: Function Optimization

Type	Severity	Location
Gas Optimization	Informational	PriceOracle.sol L918-L973

Description:

The `setExchangeRate()` function can be optimized in two parts, hence saving gas.

Recommendation:

We advise to store the `ExchangeRateModel(exchangeRateModel)` into a local variable instead of casting the `address` to `ExchangeRateModel`. Also, introduce a `storage` variable and update that instead of redundantly looking-up the `exchangeRates` mapping for the specific `asset`.

Alleviation:

The recommendation was not taken into account, with the dForce team stating "No plan to change the oracle implementation."

POE-10: Code Optimization

Type	Severity	Location
Gas Optimization	Informational	PriceOracle.sol L1149

Description:

The `for` loop conditional redundantly does a look-up to the `length` member of the `_assets` array on every iteration.

Recommendation:

We advise to introduce a local variable with the value of `_assets.length` instead.

Alleviation:

The recommendation was not taken into account, with the dForce team stating "No plan to change the oracle implementation."

POE-11: Ambiguous NetSpec Comments

Type	Severity	Location
Coding Style	Informational	PriceOracle.sol L1248-L1252

Description:

The linked NatSpec comments are describing some `return` values twice and are missing the boolean `return` value description.

Recommendation:

We advise to update the linked NatSpec comments.

Alleviation:

The recommendation was applied in commit 397ec65b5676bd2f64d72e532e961595ab931e3d.

POE-12: Introduction of `require` Statements

Type	Severity	Location
Volatile Code	Minor	PriceOracle.sol L401, L888, L1344, L1522, L1526, L1579

Description:

A failed `assert` statement will consume all the gas available to the call.

Recommendation:

We advise to change the linked `assert` statements with `require` ones.

Alleviation:

The recommendation was not taken into account, with the dForce team stating "No plan to change the oracle implementation."

POE-13: Function Visibility Optimization

Type	Severity	Location
Gas Optimization	● Informational	PriceOracle.sol L683, L797, L822, L854, L887, L918, L982, L1035, L1240, L1254, L1305, L1600

Description:

The linked functions are declared as `public`, yet they are never called by the contract.

Recommendation:

We advise that the functions' visibility specifiers are set to `external`, optimizing the gas cost of the function.

Alleviation:

The recommendation was not taken into account, with the dForce team stating "No plan to change the oracle implementation."

POE-14: Naming Conventions

Type	Severity	Location
Coding Style	● Informational	PriceOracle.sol L259, L262, L268, L269, L498, L500, L565, L683, L797, L822, L854, L887, L1078, L1112, L1141

Description:

The linked `public` / `external` variables and functions do not follow the Solidity standards in regards to their naming.

Recommendation:

We advise to closely follow the Solidity style guide.

Alleviation:

The recommendation was not taken into account, with the dForce team stating "No plan to change the oracle implementation."

POE-15: Inexistant Input Sanitization

Type	Severity	Location
Volatile Code	Informational	PriceOracle.sol L822-L847

Description:

The `_setPendingAnchorAdmin()` function does not check whether the input value is equal to the existing one.

Recommendation:

We advise to add a `require` statement checking against the existing value of `pendingAnchorAdmin`.

Alleviation:

The recommendation was not taken into account, with the dForce team stating "No plan to change the oracle implementation."

ETH-01: Ambiguous Statement

Type	Severity	Location
Volatile Code	Informational	iETH.sol L37

Description:

The linked statement does not properly use the `_spender` parameter. Also, the `_doTransferIn()` function does not follow the functionality explained in the NatSpec comments.

Recommendation:

We advise to revise the `_doTransferIn()` function.

Alleviation:

The recommendation was not taken into account.

ETH-02: Usage of `transfer()` for sending Ether

Type	Severity	Location
Volatile Code	Minor	iETH.sol L49, L124, L139

Description:

After EIP-1884 was included in the Istanbul hard fork, it is not recommended to use `.transfer()` or `.send()` for transferring ether as these functions have a hard-coded value for gas costs making them obsolete as they are forwarding a fixed amount of gas, specifically `2300`. This can cause issues in case the linked statements are meant to be able to transfer funds to other contracts instead of EOAs.

Recommendation:

We advise that the linked `.transfer()` and `.send()` calls are substituted with the utilization of the `sendValue()` function from the `Address.sol` implementation of OpenZeppelin either by directly importing the library or copying the linked code.

Alleviation:

The recommendation was not taken into account, with the dForce team stating "No plan to change, aiming to restrict the gas."

ETH-04: Non Standard Contract Naming

Type	Severity	Location
Coding Style	Informational	iETH.sol L11

Description:

The contract naming does not follow the Solidity naming conventions.

Recommendation:

We advise to closely follow the Solidity style guide.

Alleviation:

The recommendation was not taken into account, with the dForce team stating "No plan to change the token contract name."

ETH-05: Inexistent Input Sanitization

Type	Severity	Location
Volatile Code	Minor	iETH.sol L186

Description:

The `receive()` function allows the contract to receive `ETH`, to repay a successful flash loan.

Recommendation:

We advise to add a `require` statement ensuring that only a contract is able to send `ETH` to the contract.

Alleviation:

The recommendation was applied in commit 397ec65b5676bd2f64d72e532e961595ab931e3d.

ETH-06: Contract Size

Type	Severity	Location
Compiler Error	Informational	iETH.sol General

Description:

Contract size exceeds byte limit, may cause an issue in mainnet deployment.

Recommendation:

No recommendation.

Alleviation:

The dForce team stated "With 200 runs of optimization, code size is okay."

ITO-02: Redundant casting to `uint8`

Type	Severity	Location
Gas Optimization	Informational	iToken.sol L34

Description:

The aforementioned line performs redundant casting to `uint8` as the value returns by the function `decimals` is already a `uint8`.

Recommendation:

We advise to remove the redundant casting to `uint8` to save gas cost associated with the casting operation.

Alleviation:

The recommendation was applied in commit 397ec65b5676bd2f64d72e532e961595ab931e3d.

ITO-03: Contract name does not comply with the convention

Type	Severity	Location
Language Specific	Informational	iToken.sol L13

Description:

The contract name `iToken` starts with the small letter, which is against the convention of the contract names as the convention is to start the contract name with capital letter.

Recommendation:

We advise to changed the name of contract to comply with the convention of contract names.

Alleviation:

The recommendation was not taken into account, with the dForce team stating "No plan to change the token contract name."

BAS-01: Possibility of replay attack in `permit`

Type	Severity	Location
Volatile Code	Minor	Base.sol L577, L45

Description:

The `permit` function on `L577` performs the operation of deriving signer address from the signature values of `v`, `r` and `s`. The state varible `DOMAIN_SEPARATOR` that is used to calculate hash has a value of `chainid` that is derived only once in initialize function, which does not change after contract deployment. The issue arises in the event of fork when the cross-chain replay attacks can be executed.
The attack scenario can be thought of as if a fork of Ethereum happens and two different networks have id of for example `1` and `9`. The chainid coded in `DOMAIN_SEPARATOR` will be the same on contracts residing in both of the forks. If the chainid `1` is stored in the contract then the `permit` transaction signed for chainid `1` will be executable on both of the forks.

Recommendation:

We advise to construct the `DOMAIN_SEPRATOR` hash inside the permit function so the current `chainid` could be fetched and only the transactions signed for current network could succeed.

Alleviation:

The recommendation was not taken into account, with the dForce team stating "No plan to change."

RDR-01: Potential for re-entrancy attacks in `claimReward`

Type	Severity	Location
Volatile Code	● Medium	contracts/RewardDistributor.sol L419-L420

Description:

The public `claimReward` function in the `RewardDistributor` contract has the potential for re-entrancy attacks due to the lack of access restriction and transferring from the arbitrary `rewardToken` address state variable to arbitrary `_holders` addresses. In the case that the caller supplies a valid account address in the `_holders` address array parameter with a non-zero `reward` value, a malicious `rewardToken` contract or `_account` address could re-enter the `claimReward` function and drain the funds, because each account's reward amount is not updated in the public `reward` address-to-amount mapping state variable until L420, following the transfer on L419.

Recommendation:

This can be resolved by following the Check-Effects-Interactions pattern, setting the `reward` value for the current `_account` to zero before the transfer by effectively swapping L419 and L420.

Alleviation:

The recommendation was applied in commit 397ec65b5676bd2f64d72e532e961595ab931e3d.

RDR-02: Missing event for updating global distribution speed

Type	Severity	Location
Implementation	Minor	contracts/RewardDistributor.sol L172

Description:

The public `_setGlobalDistributionSpeed` function in the `RewardDistributor` contract allows the owner to modify the `globalDistributionSpeed` state variable without emitting an event, which makes it difficult to track off-chain.

Recommendation:

Consider introducing a `SetGlobalDistributionSpeed` event in order to safely track changing of the `globalDistributionSpeed` state variable on chain.

Alleviation:

The recommendation was applied in commit 397ec65b5676bd2f64d72e532e961595ab931e3d.

RDR-03: Lack of address check in `_updateReward`

Type	Severity	Location
Volatile Code	Minor	contracts/RewardDistributor.sol L360

Description:

The internal `_updateReward` function in the `RewardDistributor` contract does not check if the supplied `_account` address parameter is non-zero.

Recommendation:

Consider introducing a requirement in order to verify that the supplied `_account` address parameter is non-zero before using it in the function.

Alleviation:

The recommendation was applied in commit 397ec65b5676bd2f64d72e532e961595ab931e3d.

RDR-04: Unnecessary underscore prefixing `_setRewardToken`

Type	Severity	Location
Naming Conventions	Informational	contracts/RewardDistributor.sol L90

Description:

The external `_setRewardToken` function in the `RewardDistributor` contract is prefixed with an underscore (`_`), which is a convention typically reserved for private and internal declarations.

Recommendation:

Since the function is external, consider renaming the function to `setRewardToken` .

Alleviation:

The recommendation was not taken into account, with the dForce team stating "Public functions starts with _ are owner functions for easy interaction on Remix or Etherscan."

RDR-05: Unnecessary underscore prefixing `_addRecipient`

Type	Severity	Location
Naming Conventions	Informational	contracts/RewardDistributor.sol L110

Description:

The external `_addRecipient` function in the `RewardDistributor` contract is prefixed with an underscore (`_`), which is a convention typically reserved for private and internal declarations.

Recommendation:

Since the function is external, consider renaming the function to `addRecipient` .

Alleviation:

The recommendation was not taken into account, with the dForce team stating "Public functions starts with _ are owner functions for easy interaction on Remix or Etherscan."

RDR-06: Unnecessary underscore prefixing `_pause`

Type	Severity	Location
Naming Conventions	Informational	contracts/RewardDistributor.sol L132

Description:

The external `_pause` function in the `RewardDistributor` contract is prefixed with an underscore (`_`), which is a convention typically reserved for private and internal declarations.

Recommendation:

Since the function is external, consider renaming the function to `pause` .

Alleviation:

The recommendation was not taken into account, with the dForce team stating "Public functions starts with _ are owner functions for easy interaction on Remix or Etherscan."

RDR-07: Unnecessary underscore prefixing `_unpause`

Type	Severity	Location
Naming Conventions	Informational	contracts/RewardDistributor.sol L144

Description:

The external `_unpause` function in the `RewardDistributor` contract is prefixed with an underscore (`_`), which is a convention typically reserved for private and internal declarations.

Recommendation:

Since the function is external, consider renaming the function to `unpause` .

Alleviation:

The recommendation was not taken into account, with the dForce team stating "Public functions starts with _ are owner functions for easy interaction on Remix or Etherscan."

RDR-08: Unnecessary underscore prefixing `_setGlobalDistributionSpeed`

Type	Severity	Location
Naming Conventions	Informational	contracts/RewardDistributor.sol L165

Description:

The public `_setGlobalDistributionSpeed` function in the `RewardDistributor` contract is prefixed with an underscore (`_`), which is a convention typically reserved for private and internal declarations.

Recommendation:

Since the function is public, consider renaming the function to `setGlobalDistributionSpeed`.

Alleviation:

The recommendation was not taken into account, with the dForce team stating "Public functions starts with _ are owner functions for easy interaction on Remix or Etherscan."

RDR-09: Inefficient early return in `updateDistributionSpeed`

Type	Severity	Location
Gas Optimization	Informational	contracts/RewardDistributor.sol L185-L187

Description:

The public `updateDistributionSpeed` function in the `RewardDistributor` contract checks if the `paused` state variable is set before returning, which is inefficient.

Recommendation:

This should most likely revert instead so that the gas is refunded.

Alleviation:

The recommendation was applied in commit 397ec65b5676bd2f64d72e532e961595ab931e3d.

RDR-10: Unnecessary underscore prefixing `_setDistributionFactors`

Type	Severity	Location
Naming Conventions	Informational	contracts/RewardDistributor.sol L271

Description:

The external `_setDistributionFactors` function in the `RewardDistributor` contract is prefixed with an underscore (`_`), which is a convention typically reserved for private and internal declarations.

Recommendation:

Since the function is external, consider renaming the function to `setDistributionFactors` .

Alleviation:

The recommendation was not taken into account, with the dForce team stating "Public functions starts with _ are owner functions for easy interaction on Remix or Etherscan."

RDR-13: `claimAllReward` should be declared external

Type	Severity	Location
Implementation	Informational	contracts/RewardDistributor.sol L429

Description:

The public `claimAllReward` function in the `RewardDistributor` contract should be re-declared as external.

Alleviation:

The recommendation was applied in commit 397ec65b5676bd2f64d72e532e961595ab931e3d.

IRM-01: `getBorrowRate` should be declared external

Type	Severity	Location
Implementation	Informational	contracts/InterestRateModel/InterestRateModel.sol L97-L101

Description:

The public `getBorrowRate` view function in the `InterestRateModel` contract should be re-declared as external.

Alleviation:

The recommendation was not taken into account, with the dForce team stating "The Interest Rate Model will be completely rewrote."

TAN-01: Unnecessary underscore prefixing `_setController`

Type	Severity	Location
Naming Conventions	Informational	contracts/TokenBase/TokenAdmin.sol L28

Description:

The external `_setController` function in the `TokenAdmin` contract is prefixed with an underscore (`_`), which is a convention typically reserved for private and internal declarations.

Recommendation:

Since the function is external, consider renaming the function to `setController` .

Alleviation:

The recommendation was not taken into account, with the dForce team stating "Public functions starts with _ are owner functions for easy interaction on Remix or Etherscan."

TAN-02: Unnecessary underscore prefixing `_setInterestRateModel`

Type	Severity	Location
Naming Conventions	Informational	contracts/TokenBase/TokenAdmin.sol L50

Description:

The external `_setInterestRateModel` function in the `TokenAdmin` contract is prefixed with an underscore (`_`), which is a convention typically reserved for private and internal declarations.

Recommendation:

Since the function is external, consider renaming the function to `setInterestRateModel` .

Alleviation:

The recommendation was not taken into account, with the dForce team stating "Public functions starts with _ are owner functions for easy interaction on Remix or Etherscan."

TAN-03: Unnecessary underscore prefixing `_setNewReserveRatio`

Type	Severity	Location
Naming Conventions	Informational	contracts/TokenBase/TokenAdmin.sol L71

Description:

The external `_setNewReserveRatio` function in the `TokenAdmin` contract is prefixed with an underscore (`_`), which is a convention typically reserved for private and internal declarations.

Recommendation:

Since the function is external, consider renaming the function to `setNewReserveRatio` .

Alleviation:

The recommendation was not taken into account, with the dForce team stating "Public functions starts with _ are owner functions for easy interaction on Remix or Etherscan."

TAN-04: Unnecessary underscore prefixing `_setNewFlashloanFeeRatio`

Type	Severity	Location
Naming Conventions	Informational	contracts/TokenBase/TokenAdmin.sol L94

Description:

The external `_setNewFlashloanFeeRatio` function in the `TokenAdmin` contract is prefixed with an underscore (`_`), which is a convention typically reserved for private and internal declarations.

Recommendation:

Since the function is external, consider renaming the function to `setNewFlashloanFeeRatio` .

Alleviation:

The recommendation was not taken into account, with the dForce team stating "Public functions starts with _ are owner functions for easy interaction on Remix or Etherscan."

TAN-05: Unnecessary underscore prefixing `_setNewProtocolFeeRatio`

Type	Severity	Location
Naming Conventions	Informational	contracts/TokenBase/TokenAdmin.sol L117

Description:

The external `_setNewProtocolFeeRatio` function in the `TokenAdmin` contract is prefixed with an underscore (`_`), which is a convention typically reserved for private and internal declarations.

Recommendation:

Since the function is external, consider renaming the function to `setNewProtocolFeeRatio` .

Alleviation:

The recommendation was not taken into account, with the dForce team stating "Public functions starts with _ are owner functions for easy interaction on Remix or Etherscan."

TAN-06: Unnecessary underscore prefixing `_withdrawReserves`

Type	Severity	Location
Naming Conventions	Informational	contracts/TokenBase/TokenAdmin.sol L142

Description:

The external `_withdrawReserves` function in the `TokenAdmin` contract is prefixed with an underscore (`_`), which is a convention typically reserved for private and internal declarations.

Recommendation:

Since the function is external, consider renaming the function to `withdrawReserves` .

Alleviation:

The recommendation was not taken into account, with the dForce team stating "Public functions starts with _ are owner functions for easy interaction on Remix or Etherscan."

TER-01: Potential for re-entrancy attacks in `_transferTokens`

Type	Severity	Location
Volatile Code	Minor	contracts/TokenBase/TokenERC20.sol L34

Description:

The internal `_transferTokens` function in the `TokenERC20` contract has the potential for re-entrancy attacks due to being accessible from the public `transfer` and `transferFrom` functions and making a call to the internal `ERC20._transfer` function.

Recommendation:

Consider utilizing the `nonReentrant` modifier on the internal `_transferTokens` function in order to protect against re-entrancy attacks.

Alleviation:

The recommendation was applied in commit 397ec65b5676bd2f64d72e532e961595ab931e3d.

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.a