DcentraLab Diligence

dcentralab.com/diligence

DOUGH.

Audit Report

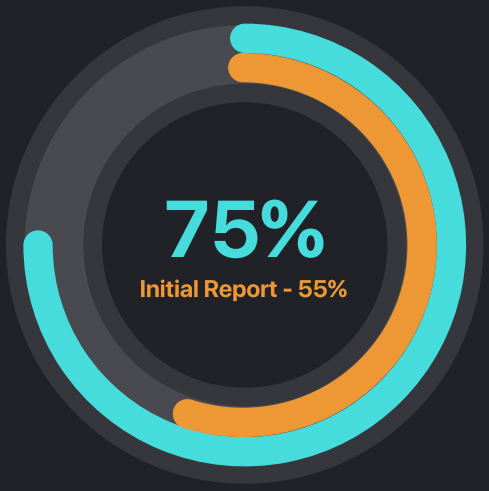
Dough Finance

https://doughfinance.xyz

Provided By DcentraLab Diligence on November 08, 2023

Security Audit Score

DcentraLab Diligence team has conducted an extensive audit on Dough Finance Contracts and has found the code to be in low risk level given proper deployment and multi-sig permissioning.



- Minimal Risk
- Small Risk
- Medium Risk
- High Risk

Scope

Audited Repository:

https://github.com/purevolcano106/DoughV1_contracts

Audited Branch:

main

Audited Commit Hash:

6940a3bd20c9c17d6164b760da13a2c9ef1dcf22

Fixes Commit Hash:

1808f1190eace4f492023135e1c533933f1d35c2

Contracts Reviewed:

- connectors/ConnectorV1AaveV3.sol
- connectors/ConnectorV1Dsa.sol
- connectors/ConnectorV1Flashloan.sol
- connectors/ConnectorV1UniswapV2.sol
- DoughV1Dsa.sol
- DoughV1Index.sol
- Interfaces.sol

Nomenclature Of Issues / Legend:

- E - Environmental - Related to repository / not directly related to the contracts
- G - Global
- A - Local, Contract DoughV1Dsa
- B - Local, Contract DoughV1Index
- C - Local, Connector contracts
    - C1 - ConnectorV1AaveV3
    - C2 - ConnectorV1Dsa
    - C3 - ConnectorV1Flashloan
    - C4 - ConnectorV1UniswapV2

Contracts Architecture Overview

Disclaimer

This audit scope was limited to the Dough contracts, but it should be noted that there might be hidden assumptions (either logical or economical) and/or latent dependencies on the external DEFI protocols being interacted with via the connector contracts (e.g. AAVE, Uniswap etc..). These interdependencies with external DEFI protocols might give rise to latent attack surfaces and exploitable vulnerabilities, and we suggest deeper auditing of the 3rd party protocols themselves in context of their potential risk factors when interacted with by the Dough contracts.

Contracts Architecture

All contracts take setting values from the DoughV1Index.sol.

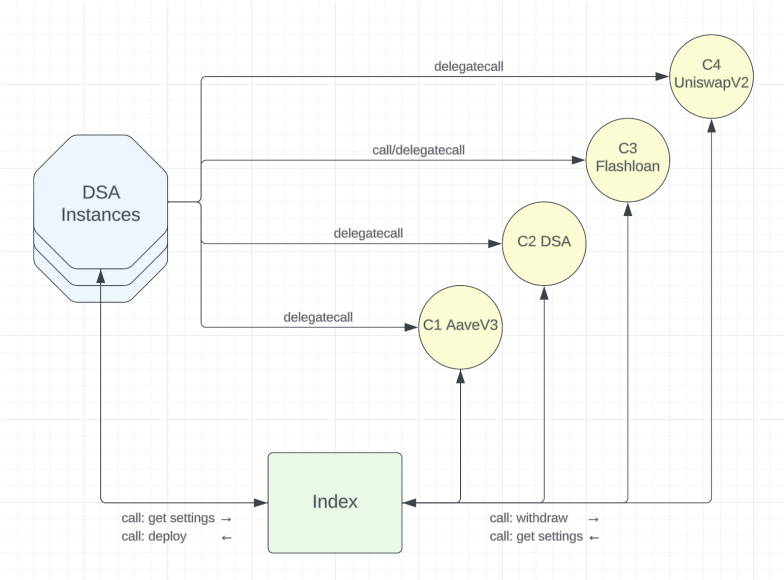
All of the connector addresses are registered inside DoughV1Index.sol.

DoughV1Dsa is a digital-smart-account which represents an account abstraction which allows users to perform DeFi related actions through an automated flow.
DoughV1Dsa serves as a multi-proxy which interacts with 4 of the connectors that currently exist, though the number of connectors is not limited on the DoughV1Index contract.

Connectors interact with protocols Uniswap and Aave and enable other simpler functionalities such as wrapping/unwrapping ETH and transfers of different kinds.

DoughV1Index is the main contract privilege wise. It is controlled by an 'owner' wallet which belongs to the Dough organization.

See architecture diagram for a proper visualization of architecture and interactions.



Dough Smart-Contract Architecture

Contracts Architecture Overview

Interactions Between The Contracts

DoughV1Dsa is a multi-proxy which interacts with other contracts using 'call' and 'delegatecall' opcodes. 'delegatecall' is used to interact with the connectors, to extend the proxy logic, and 'call' is used to interact with the index and the flash loan connector, to utilize them as an external caller.

Index contract is called by all other contracts via 'call', to get the settings they need.

Withdraw Tokens Function Of Connectors

All connectors have one function in common which is unrelated to the proxy flow, it is called 'withdrawToken'. This function is present on the contract in case that tokens accidentally end up on the connector contract, then the owner of the Index which is a trustworthy Dough organization wallet may withdraw stuck tokens.

Shield Functionality Of DoughV1Index

AaveV3 pools have health levels computed for each user, shield data set on index contract for a specific dsa will result in DSA allowing the shield executor to call 'deloop' if health level retrieved from the pool is below the threshold of the shield, this action shall result in an increase of user's health factor value and this topic is elaborated through the findings of this report. This is what the '_shield_dsa_hf_trigger' parameter of the shield serves for.

Shield '_shield_dsa_prev' and '_shield_dsa_next' mappings serve the purpose of navigating as in a double linked list.

Four present mappings of the shield together represent a structure which acts as a node of a double linked list of active shields. Active shields are sorted by the age of activation, ascending.

As we have already mentioned, two mappings serve for navigation through the list, in addition to that statement, other two mappings contain shield data. Only one of these two is being utilized in the DSA logic.

'shield_dsa_cnt' is the count of total active shields on dsa - the practical use of this value is not present on contracts side.

'_shield_dsa_hf_target' parameter of the shield remains unused and its purpose is unclear.

This structure can be optimized, as the context of list-like behavior and '_shield_dsa_hf_target' parameter are unknown. This is further elaborated in the findings of this report.

Contracts Architecture Overview

Shield Executor

We see this entity as an automated backend process which checks for the user's health factor and calls for 'deloop' if it is below threshold. This applies for when the user's shield is active.

Flashloan Flow

Flashloan action is being initiated at the user's DSA instance. DSA calls the flashloan connector via delegatecall, using it as a piece of its own logic.

Logic of the flashloan connector contains, except for the DSA (proxy/router) dedicated logic, an external call to itself which makes it both a piece of DSA implementation and a standalone contract (this is visible in the flashloan diagram below in steps 1. and 2.).

Flashloan connector, now acting as a standalone contract, is saving 'dataHash' in it's storage (to be able to checksum the call data coming from AaveV3 Pool in step 4.), and proceeds to make step 3., calling 'flashLoanSimple()' function on the Aave pool.

To ease the use of the Aave pool flashloan functionality, connector contract inherits 'FlashLoanSimpleReceiverBase'. Conventionally, upon the 'flashLoanSimple' call, Aave pool is calling back 'executeOperation' function on the flashloan connector.

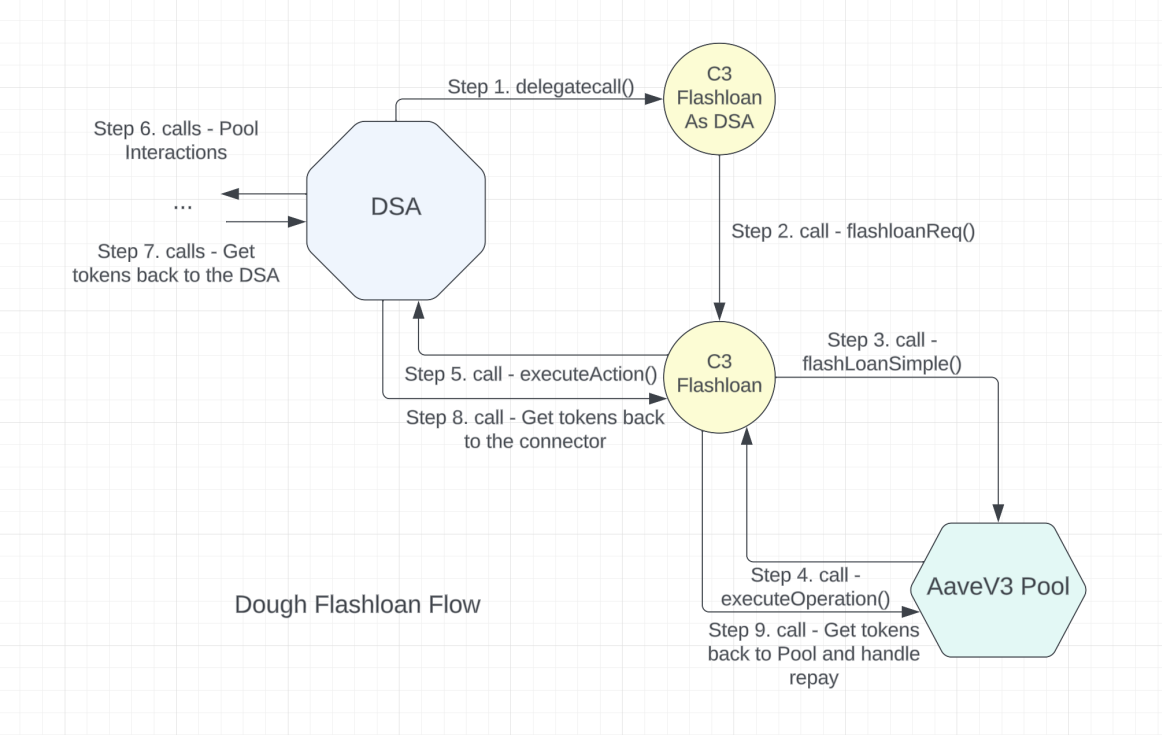
This results in connector managing the fees and calling 'executeAction' on the DSA. Using the flashloan system in this integration there are 2 possible options for the DSA owner, to either 'loop' or 'deloop' using flashloan funds.

Looping is a combination of supply and borrow, while delooping is a combination of repay and withdraw. Mentioned actions are being performed with DSA on the Aave pool.

After the goal of the flashloan has been achieved, the call stack is returned back to the pool, moving funds from DSA to the connector and then finally to the pool itself where repayment logic is being handled.

This process includes management of multiple fee payments, Aave premiums, Dough flashloan fee and optional Dough shield executor fee (if shield is calling for 'deloop').

Contracts Architecture Overview



General Notices

Risks:

DcentraLab Diligence (DD) has performed all checks and verifications in its capacity to ascertain the safety of the code. However, it should be noted that misuse of the code, bad deployment practices, bad key management, exposing of private keys of the deployer and/or owner address and/or multi-sig signer addresses and/or fee collector address and/or any exposition of the code to malicious actors may result in an exploit of the code and loss of state and/or funds. Furthermore, there is always a chance that other Smart Contracts code could be written and deployed to cause the provided code by DD to act outside the intended scope by the client, to the point of causing state corruption or loss of funds to the client of the users of the code.

Issues Severity Reference Table

Type

Discussion

The issue severity is dependent on design, centralization, and product specifications of the project.

Informational

This issue is not critical and does not pose an immediate threat to the functionality or security of the smart contract. It is simply an informational item that the auditors have identified and recommends addressing for best practices or to improve the overall performance of the contract.

Low

This issue is relatively minor and does not pose a significant risk to the functionality or security of the smart contract. While it is recommended to address these issues to ensure the highest level of quality and security, they are not likely to cause significant problems if left unaddressed.

Medium

This issue poses a moderate risk to the functionality or security of the smart contract. While it may not be immediately exploitable, it has the potential to cause problems in the future if left unaddressed. It is recommended to address these issues as soon as possible to prevent any potential negative impact on the contract.

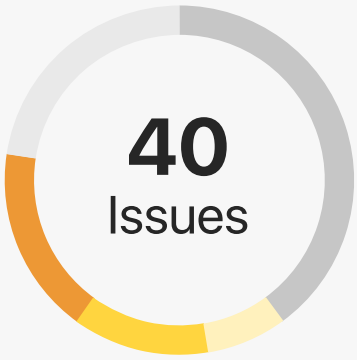
High

This issue poses a significant risk to the functionality or security of the smart contract. Addressing these issues as soon as possible is recommended to prevent any potential negative impact on the contract. Failure to address these issues could result in significant problems and potential loss of funds or other assets.

Critical

This issue poses an immediate and severe risk to the functionality or security of the smart contract. It is recommended to address these issues immediately to prevent any potential negative impact on the contract. Failure to address these issues could result in catastrophic problems and significant loss of funds or other assets.

Findings Summary



- Discussion
- Informational
- Low Risk
- Medium Risk
- High Risk
- Critical Risk

ID	Title	Severity	Status
E.1	Typographical errors	Informational	Resolved
E.2	'close deloop' test is missing conditioning	Low	Resolved
E.3	Insufficient test coverage	High	Resolved
G.1	Missing code styling conventions	Informational	Partially Resolved
G.2	Use of 'require' statements with solidity compiler version ^0.8.4	Informational	Partially Resolved
G.3	Storage layout	High	Resolved
G.4	Presence of unused constants	Informational	Resolved
G.5	Presence of constants declared as variables	Informational	Resolved
G.6	Duplicate constant values	Informational	Resolved
G.7	SafeERC20 imported but not used	High	Resolved

Findings Summary

ID	Title	Severity	Status
G.8	Unused library	Informational	Resolved
G.9	Missing zero check in constructors	Low	Resolved
G.10	Interfaces.sol import management	Informational	Resolved
G.11	Lack of flexibility in the architecture	Discussion	Acknowledged
G.12	Fixed interest rate mode on borrows/repays	Discussion	Acknowledged
G.13	Impossible to make a repay when 'opt' equals 'true'	High	Resolved
G.14	Possible miscalculations on 'supply' flow	Medium	Resolved
G.15	Centralization risk of shield executor entity	Discussion	Acknowledged
A.1	Unused 'onlyOwner' modifier	Informational	Resolved
A.2	Lack of zero checks in constructor	Low	Resolved
A.3	DSA Upgradeability	Discussion	Acknowledged
B.1	Ensure secure ownership of the contract	Discussion	Acknowledged
B.2	Enable Upgradeability	Discussion	Acknowledged
B.3	Lack of fee value checks in setters	Medium	Resolved

Findings Summary

ID	Title	Severity	Status
B.4	Shield structure	Informational	Resolved
B.5	Double linked list for shields	Discussion	Acknowledged
B.6	Owner can frontrun transactions and set unexpected high fees	Medium	Resolved
C1.1	Redundant 'else if' statement	Informational	Resolved
C1.2	Repeating hardcoded value '10000' can be turned into a constant	Informational	Resolved
C2.1	Redundant 'else if' statement	Informational	Resolved
C2.2	Unintentional shield executor activation and possible fund loss	High	Resolved
C3.1	Missing order in contract code	Informational	Resolved
C3.2	Redundant cast to 'bytes32'	Informational	Resolved
C3.3	Unification of fees in flashloan flow	Discussion	Resolved
C3.4	Swapping may lead to unexpected losses	High	Resolved
C3.5	Gas spendings optimization	Informational	Resolved
C3.6	Flashloan utilized amount volatility	Medium	Resolved

Findings Summary

ID	Title	Severity	Status
C4.1	Loss of precision on amount of tokens being swapped	Medium	Resolved
C4.2	Zero value of minAmountOut	High	Resolved
C4.3	Visibility of swap result	Discussion	Resolved

Complete Analysis

Environmental Issues:

ID E.1: **Status: Resolved**

Informational | Typographical errors

Present at: test/dsa.ts & scripts/deploy_sepolia.ts

Description: In multiple places the word 'connector' is misspelled as 'coonector'.

Recommendation: Consider fixing misspelled words.

ID E.2: **Status: Resolved**

Low | 'close deloop' test is missing conditioning

Present at: test/dsa.ts @ L571-579

Description: Test is supposed to close the user's existing debt using 'opt' argument, which is not accomplished by the function call.

Recommendation: Implement conditioning so that, if the user's debt is not closed, the test will fail and indicate the error in contract logic.

Complete Analysis

ID E.3: **Status: Resolved**

High | Insufficient test coverage

Present at: test/dsa.ts

Description: Tests are written in such a way that does not prove the validity of the contract logic and they do not cover all of the desired functionalities.

Recommendation: Make sure to write detailed tests with proper conditions in order to ensure maximum safety of your code before going to production. It is of greatest importance that your team tests the code thoroughly, both through the unit tests and manual testing, in order to reduce the chance of a vulnerability or a logical error occurrence.

Complete Analysis

Global Issues:

ID G.1: **Status: Partially Resolved**

Informational | Missing code styling conventions

Present at: Contracts directory / DoughV1Dsa + Connectors

Description: Throughout the codebase, variables and function names are written in different cases which results in poor readability.

Recommendation: Consider applying a single convention to all of the variables and function names.

Auditor Comment: Proper styling conventions were applied partially in the dedicated fix commit and during further fixes new data items with improper styling were introduced.

ID G.2: **Status: Partially Resolved**

Informational | Use of 'require' statements with solidity compiler version ^0.8.4

Present at: Contracts directory / DoughV1Dsa + Connectors

Description: Your chosen compiler version is above 0.8.4 (version that introduced 'custom errors'), this implies that you can reduce gas consumption and bytecode size by introducing them instead of 'require' statements. After introduction of 'custom errors', the 'require' statements are considered redundant as there is no additional value that they provide.

Helpful Resource: https://soliditylang.org/blog/2021/04/21/custom-errors/

Recommendation: Replace 'require' statements with 'custom errors'.

Complete Analysis

ID G.3: **Status: Resolved**

High | Storage layout

Present at: Contracts directory / DoughV1Dsa + Connector contracts

Description: With DoughV1Dsa -> connectors 'delegatecall' interaction taking place, storage layout is copied but so are the constants, since constants are not a part of the contract storage you can remove them from the places where they're not used.
There are variables which can become constants as they are hard-coded but not declared as such. If you mark every constant value with the keyword 'constant' formally, those values will not take place in the contract's storage.
Once this is cleared out of the way, Dough connectors are left with 2 variables: owner and DoughV1Index.
Those 2 variables represent the storage layout to be respected among the connectors as they reflect storage of the main dsa contract.

Recommendation: Consider redeclaring hard-coded variables as constants and removing constants from places where they're not used, as they are not part of the storage layout.

Connection: This issue is strongly connected with the following issues: G.4, G.5 and G.6.

ID G.4: **Status: Resolved**

Informational | Presence of unused constants

Present at: Contracts directory / DoughV1Dsa + connectors

Description: Mentioned contracts contain the same pattern of constants, among which many are unused.

Recommendation: Consider removing the unused constants in each contract.

Complete Analysis

ID G.5: **Status: Resolved**

Informational | Presence of constants declared as variables

Present at: Contracts directory / DoughV1Dsa + connectors

Description: Mentioned contracts contain constants / unchangeable values declared as variables. Variables are the following:

- aave_v3_pool
- aave_v3_data_provider
- uniswap_v2_router

Recommendation: Consider declaring mentioned variables as constants instead.

ID G.6: **Status: Resolved**

Informational | Duplicate constant values

Present at: Contracts directory / DoughV1Dsa + Connectors

Description: Mentioned contracts contain constant values which repeat. Such as UNISWAP_V2_ROUTER and uniswap_v2_router. You might want to declare uniswap_v2_router along with providing an address directly and casting the interface. Constant pairs this optimization can be applied to are the following:

- UNISWAP_V2_ROUTER & uniswap_v2_router
- AAVE_V3_DATA_PROVIDER & aave_v3_data_provider
- AAVE_V3_POOL & aave_v3_pool

Note that the raw address of the interface can still be accessed via address(<INTERFACE INSTANCE>).

Complete Analysis

Example:
Redefine this statement:
address private constant UNISWAP_V2_ROUTER = 0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D;
IUniswapV2Router private uniswap_v2_router = IUniswapV2Router(UNISWAP_V2_ROUTER);

Like this:
IUniswapV2Router private constant UNISWAP_V2_ROUTER = IUniswapV2Router(0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D);

If you need an address in the pure form you can access it via 'address(UNISWAP_V2_ROUTER)'.

Recommendation: Consider merging mentioned duplicate values into a single constant in a described way.

Global Issues:

ID G.7: **Status: Resolved**

High | SafeERC20 imported but not used

Present at: Contracts directory / DoughV1Dsa + Connectors

Description: SafeERC20 library has been imported but never used. It is of great importance that contracts which interact with multiple tokens utilize SafeERC20 properly, otherwise issues with token interactions may occur.

Recommendation: Utilize SafeERC20 in the transfer related flows. Replace each 'transfer' with 'safeTransfer' and 'transferFrom' with 'safeTransferFrom'.

Complete Analysis

ID G.8: **Status: Resolved**

Informational | Unused library

Present at: Contracts directory / DoughV1Dsa + Connectors

Description: SafeMath library which is imported into the DoughV1Dsa has never been used.

Recommendation: Consider removing unused libraries. Notice: SafeMath is not needed starting Solidity version 0.8.0 as arithmetic operations revert to underflow and overflow.

ID G.9: **Status: Resolved**

Low | Missing zero check in constructors

Present at: Contracts directory / Connectors

Description: doughV1Index which is the only argument in the connector constructors is not being checked against the zero address which might lead to a bad contract initialization.

Recommendation: Consider adding zero check for the mentioned argument.

ID G.10: **Status: Resolved**

Informational | Interfaces.sol import management

Present at: Contracts directory / DoughV1Dsa + Connectors

Description: Mentioned Dough contracts import Interfaces.sol file, which contains multiple interfaces. From this statement it is not clear which ones are used, this reduces code readability.

Recommendation: Consider increasing readability by importing only interfaces which are used by the contract using the "import {InterfaceA, InterfaceB} from './interfaces.sol'" scheme.

Complete Analysis

ID G.11: **Status:** Acknowledged

Discussion | Lack of flexibility in the architecture

Present at: Contracts directory

Recommendation: After careful review of the Dough smart-contract architecture we noticed the lack of flexibility. Multiple values are hardcoded, lacking settings and the core Dsa contract is not upgradeable. This can push user's towards migration to new versions or different protocols or an EOA which is a fairly exhausting process. Consider increasing architectural flexibility and therefore provide better UX.

Related issues: G.12, A.4, B.2.

Project Response: The team may wish to add a significant number of extra protocols and favor the redeployments of contracts rather than the use of proxies and the problems associated with them.

ID G.12: **Status:** Acknowledged

Discussion | Fixed interest rate mode on borrows/repays

Present at: Contracts directory / DoughV1Dsa + AaveV3Connector

Description: Interest rate mode has a hardcoded value of 2 on each borrow and repay call. While this might be a good solution for the casual users it could be an obstacle for more advanced users.

Project Response: The model favored by the project maintains a value of 2.

Complete Analysis

ID G.13: **Status: Resolved**

High | Impossible to make a repay when 'opt' equals 'true'

Present at: Contracts directory / DoughV1Dsa + AaveV3Connector + FlashloanConnector

Description: Context of 'opt' parameter picked up from the flow is to get the exact amount that the user has to repay, in context of the flashloan flow. Once the amount to repay is retrieved, the fee is applied to it which means that the user will not repay fully. This defeats the purpose of 'opt' parameter as by using it you will never fully repay. This is applied in each flow that includes 'repay'.

Recommendation: Consider applying fees independently of the amount being used in the flow.

ID G.14: **Status: Resolved**

Medium | Possible miscalculations on 'supply' flow

Present at: Contracts directory / DoughV1Dsa + AaveV3Connector + FlashloanConnector

Description: In flows including 'supply' action, fees are being taken prior to the action itself. They are taken from the amount which is specified by the users, potentially leading them to think that they are 'supplying' a different amount. The exact amount 'supplied' will not be transparent to the user in this case.

Recommendation: Consider applying fees independently of the amount being used in the flow.

Complete Analysis

ID G.15: **Status:** Acknowledged

Discussion | Centralization risk of shield executor entity

Present at: Architecture outside of contracts

Description: Centralization risk of running the shield executor on the project servers.

Recommendation: Consider using some decentralized oracle / decentralized verifiable execution protocol, and/or enabling multiple sources to be the shield executor to open this up for more service providers and enhance security of the feature.

Project Response: We have discussed this at length and plans are already in place to use multiple shield executors in the future.

Complete Analysis

Local Issues:
DoughV1Dsa

ID A.1: **Status: Resolved**

Informational | Unused 'onlyOwner' modifier

Present at: DoughV1Dsa.sol @ L38

Description: Modifier 'onlyOwner' is defined but never used.

Recommendation: Consider either removing or utilizing the modifier.

ID A.2: **Status: Resolved**

Low | Lack of zero checks in constructor

Present at: DoughV1Dsa.sol @ L31-34

Description: The zero checks of the constructor arguments are not present.

Recommendation: Consider adding zero checks for the mentioned arguments to ensure their validity.

Complete Analysis

ID A.3: **Status:** Acknowledged

Discussion | DSA Upgradeability

Present at: DoughV1Dsa.sol

Recommendation: Currently DoughV1Dsa is a static contract and migration from it could be a fairly exhausting process for users. Consider making DoughV1Dsa upgradeable, so users can always be up to date with the latest implementation containing new features and security measures, without migrating all their funds from the original contract.

Project Response: We will consider this. As mentioned due to the potential integrations of other protocols and the issues faced with proxy contracts we will most likely keep the current setup but understand the point raised regarding upgradeability.

Complete Analysis

Local Issues:
DoughV1Index

ID B.1: **Status:** Acknowledged

Discussion | Ensure secure ownership of the contract

Present at: DoughV1Index.sol

Recommendation: To enhance security, it is crucial to ensure that the owner of the contract is a multisig wallet or a decentralized autonomous organization (DAO). This will help prevent unauthorized access over the index, maintaining the integrity of the system.

Project Response: Due to the nature of the execution of functions like Shield it could actually cause issues for the contracts not to be executed in a timely manner if reliance on multisig is present. The ownership will be designated to the organization.

Diligence Response: We consider this approach unsafe. While the shield executor should be a backend wallet, there is no need for the owner to be an ordinary wallet as it only manages settings.

ID B.2: **Status:** Acknowledged

Discussion | Enable Upgradeability

Present at: DoughV1Index.sol

Recommendation: To enhance the overall system's compatibility and maintain consistency, it is recommended to enable upgradability of the DoughV1Index contract. Since connectors are replaceable and new ones can be added too, you could end up in need to add new variables or functionalities to the existing contract. This can also improve the management of new DSA contract deployments, as you could inherit new implementations in your upgrade. That way you could keep the same index contract and update the DSA to a new version for future deployments.

Complete Analysis

Project Response: The Team May Wish To Add A Number Of Protocols In The Future And Prefer Redeployment As Opposed To Using Proxies And The Problems Associated With Them.

ID B.3: **Status: Resolved**

Medium | Lack of fee value checks in setters

Present at: DoughV1Index.sol

Description: Some of the fee values being checked are supposed to respect the precision inside of the calculations where they're applied. This means that DoughV1Index should be aware of the precision, or that precision value should exist locally in the contract and be in sync with precision being used inside the calculations. This may result in the owner setting an unexpectedly large fee by accident, which can either make user's pay a very large fee or break the transaction if there is a lack of funds to cover the expense.In order to ensure safety of users' funds, introduce a check which makes sure that fee is significantly lower than precision value in following functions:

- setSupplyFee @ L82
- setWithdrawFee @ L86
- setBorrowFee @ L90
- setRepayFee @ L94
- setSwapFee @ L98
- setFlashloanFee @ L102

Recommendation: Make sure to implement missing checks. It is important that fee values are not only respecting the precision but that they're also significantly smaller.

Related issues: B.7

Complete Analysis

ID B.4: **Status: Resolved**

Informational | Shield structure

Present at: DoughV1Index.sol

Description: Shield information is spread among multiple mappings, 'trigger', 'target', 'prev' and 'next'. In order to increase code readability and ease the management of values contained in these mappings, you can merge them into a struct and have a single mapping 'address => Shield'.

Recommendation: Consider merging mentioned mappings into a single one, pointing to a structure containing all of the needed values. Since the mappings you created act together as a double linked list, now you can interact with a single struct which serves as a node of your list more conventionally.

ID B.5: **Status: Acknowledged**

Discussion | Double linked list for shields

Present at: DoughV1Index.sol

Recommendation: Revisit the functionality and attributes of shields data structure in the form of the double linked list and see if such behavior serves proper purpose or not. If it does not serve purpose consider deducting the structure to a simpler one.

Complete Analysis

ID B.6: **Status: Resolved**

Medium | Owner can frontrun transactions and set unexpected high fees

Present at: DoughV1Index.sol

Description: Fees can be set at any point in time by the owner, which may result in frontrunning users with high fees, as fees values currently do not have any reasonable checks.

Recommendation: Consider introducing reasonable limits to the fee values in each setter, making sure that fees are not greater than a certain fraction of the precision.

Related issues: B.3

Complete Analysis

Local Issues:
ConnectorV1AaveV3

ID C1.1: **Status: Resolved**

Informational | Redundant 'else if' statement

Present at: ConnectorV1AaveV3.sol

Description: Once 'delegateDoughCall' call is made, '_actionId' is required to be '< 4'. Later during the flow once there is case segregation based on the '_actionId' it is being rechecked for a precise value. Last check in the mentioned flow: "else if (_actionId == 3)" is redundant, as every other case has already been checked.

Recommendation: Consider replacing mentioned 'else if' statement with 'else' in order to optimize gas spendings of the call.

ID C1.2: **Status: Resolved**

Informational | Repeating hardcoded value '10000' can be turned into a constant

Present at: ConnectorV1AaveV3.sol

Description: In multiple fee computations hardcoded value '10000' is present. Mentioned value can be replaced with a constant (ex. 'PRECISION') which will have the same value.

Recommendation: We recommend applying the advice from issue description as such changes can ease the further development of the contract and increase overall code readability.

Complete Analysis

Local Issues:
ConnectorV1Dsa

ID C2.1: **Status: Resolved**

Informational | Redundant 'else if' statement

Present at: ConnectorV1Dsa.sol

Description: Once 'delegateDoughCall' call is made, '_actionId' is required to be '< 4'. Later during the flow once there is case segregation based on the '_actionId' it is being rechecked for a precise value. Last check in the mentioned flow: "else if (_actionId == 3)" is redundant, as every other case has already been checked.

Recommendation: Consider replacing mentioned 'else if' statement with 'else' in order to optimize gas spendings of the call.

ID C2.2: **Status: Resolved**

High | Unintentional shield executor activation and possible fund loss

Present at: ConnectorV1Dsa.sol

Description: Based on the user's health factor value retrieved from the pool and custom set shield parameter 'shield_dsa_hf_trigger' of DoughV1Index, the shield executor is allowed to call for 'deloop'. This allows for the case when a user might 'loop', overseeing the current state and future change of the health factor, which will lead to shield executor calling for a 'deloop', resulting in a loss of user's funds due to transaction price, fees and debt penalty.

Recommendation: We recommend you to implement a check upon 'loop' action execution, seeing if the health factor is near to or below the threshold and revert if that is the case. Since the conditions of the pool and user activities may vary, this solution might not resolve the issue completely so make sure to carefully think through a solution to this.

Complete Analysis

Local Issues:
ConnectorV1Flashloan

ID C3.1: **Status: Resolved**

Informational | Missing order in contract code

Present at: ConnectorV1Flashloan.sol

Description: This contract includes functions, modifiers, variables, constants and structs which are not sorted in any particular order. This heavily impacts the code readability.

Recommendation: Consider applying conventions and enforcing order on the contract.

ID C3.2: **Status: Resolved**

Informational | Redundant cast to 'bytes32'

Present at: ConnectorV1Flashloan.sol @ L108

Description: On the mentioned line, the return value of keccak256 (which is bytes32) is being casted to bytes32.

Recommendation: Consider removing the redundant cast.

ID C3.3: **Status: Resolved**

Discussion | Unification of fees in flashloan flow

Present at: ConnectorV1Flashloan.sol

Description: During the execution of flashloan flow, Dough fees are being applied in 2 places. If the shield executioner is a company wallet, consider unifying the shield executioner fee along with the flashloan fee in the same place. This can result in cheaper transactions for users and therefore better UX.

Complete Analysis

ID C3.4: **Status: Resolved**

High | Swapping may lead to unexpected losses

Present at: ConnectorV1Flashloan.sol @ L122-139

Description: Flashloan connector works only with WETH and WBTC, shield executioner fee has a value in WETH. In order to pay the fee, flow includes a 'swapTokensForExactTokens' call, which should secure a conversion of WBTC to WETH at any price. Due to the price oscillations this may cause losses on users' side, especially as they're paying not only the fee but for the conversion flow which is impactful in terms of gas spendings.

Since 'SHIELD_EXECUTE_FEE' is a hard number of WETH, there should be some logic flow to make sure there is enough amount to swap and/or to allocate for fee, and that the fee value is sensical with the amount of tokens interacting with, one solution could be to make it a percentual value - logic has to make sure that fee is worth much less than the worth of the amount of flashloaned tokens being borrowed out in order for this operation to provide benefits to user.

Recommendation: This issue can be resolved in multiple ways, simplest one being to have fee value in WBTC too, so it can be taken directly from index - or just applying a percent based fee on each token flashloan flow is operating with. Make sure to apply one of the mentioned solutions or create one of your own in a similar manner.

ID C3.5: **Status: Resolved**

Informational | Gas spendings optimization

Present at: ConnectorV1Flashloan.sol

Description: Each time fees are being retrieved they're being transferred to other addresses, increasing the transaction cost.

Recommendation: Consider accumulating fees locally and letting an authorized wallet withdraw them all at once.

Complete Analysis

ID C3.6: **Status: Resolved**

Medium | Flashloan utilized amount volatility

Present at: ConnectorV1Flashloan.sol

Description: In order to pay multiple fees, flow is going through the operations with non-deterministic outcomes. This may cause confusion among users as they won't be able to determine with which amount they are operating with. This applies to all functionalities that are offered through the 'executeOperation' flow. This may result in users not being able to properly repay their debt, and spend additional funds making multiple transactions in order to do so.

Recommendation: Apply fees independently of the token amount working with and make sure everything is transparent to the user through the UI.

Complete Analysis

Local Issues:
ConnectorV1UniswapV2

ID C4.1: **Status: Resolved**

Medium | Loss of precision on amount of tokens being swapped

Present at: ConnectorV1UniswapV2.sol / delegateDoughCall @ L48-69

Description: Mentioned function is called with a purpose of swapping tokens using Uniswap router. As one of the arguments in the call, the user is supposed to provide the amount. Fee is subtracted from the amount in order to determine the amount of tokens which will be swapped. As a result of this flow, users might not be able to properly manage the exact amount of tokens to swap, as the amount of tokens to swap + fee is provided as function argument. Once the fee is subtracted, the user's amount of tokens to swap might not align with the expectations.

Recommendation: We consider using the fee amount separate from the swap amount as a better approach. Users should be able to provide an exact amount of tokens to swap, and the fee should be computed based on it but not be subtracted from the amount, so the final amount of tokens spent by the user will be 'amount + fee'. This way a user can always provide an amount of tokens to swap without the loss of precision. It is important to make sure that everything is transparent to the user through the UI.

ID C4.2: **Status: Resolved**

High | Zero value of minAmountOut

Present at: ConnectorV1UniswapV2.sol / delegateDoughCall @ L48-69

Description: Current flow of interaction with the Uniswap router does not allow the user to choose 'minAmountOut', furthermore this value is a fixed zero, which may result in great loss for the user.

Recommendation: It is fundamental that users have proper fund protection during swaps. They should be able to provide 'minAmountOut' themselves. You should provide a way for users to do so.

Complete Analysis

ID C4.3: **Status: Resolved**

Discussion | Visibility of swap result

Present at: ConnectorV1UniswapV2.sol / delegateDoughCall

Description: Upon successful swap users should be able to see the amount of tokens that they received.

Recommendation: Consider providing a proper notification with the result of the user's swap through the UI.

Disclaimer:

DcentraLab Diligence (DD) has provided the code to the client as is and assumes no responsibility nor legal liability for any use client may do with the code. Any and all usage and/or deployment of the code provided by DcentraLab Diligence will be done solely by the client, at the sole discretion, responsibility, risk, and legal liability of the Client, and DD will not be held accountable or liable for any loss of funds, security exploits or incidents, or any other unintended or negative outcome that may occur in relation to the code provided by DD.

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 DD 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 and the provided code or services as part of the SOW pertaining to 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 it 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. DD's position is that each company and individual are responsible for their own due diligence and continuous security. DD'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 DD are subject to dependencies and are under continuing development. You agree that your access and/or use, including but not limited to any services, code, 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, DcentraLab Diligence (DD) HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, DD SPECIFICALLY DISCLAIMS

ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM THE COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

WITHOUT LIMITING THE FOREGOING, DD MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT / VERIFICATION REPORT, WORK PRODUCT, CODE OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET THE 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 DISCLAIMER Dough Finance Contracts FOREGOING, DD PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET THE 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 DD NOR ANY OF DD'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION, CODE OR CONTENT PROVIDED THROUGH THE SERVICE. DD 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 CODE, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, CODE, 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 THE 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 THE 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 DD'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 DD WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS. THE REPRESENTATIONS AND WARRANTIES OF DD CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF THE 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 DD 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, CODE, OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL,
REGULATORY, OR OTHER ADVICE.

dcentralab.com/diligence

DcentraLab Diligence

Provided By DcentraLab Diligence on November 08, 2023