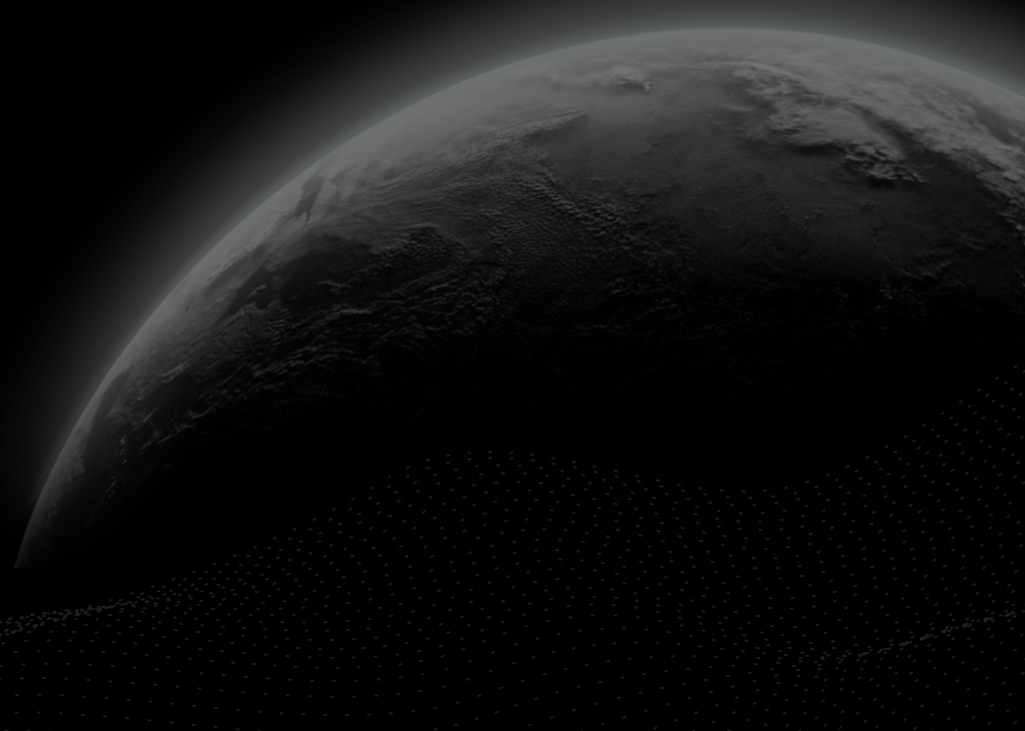
CERTIK

Security Assessment

DeFi Franc

CertiK Verified on Sept 25th, 2022



CertiK Verified on Sept 25th, 2022

DeFi Franc

The security assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES	ECOSYSTEM	METHODS
DeFi	Ethereum	Manual Review, Static Analysis

LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 09/25/2022	N/A

CODEBASE

https://bitbucket.org/grizzlyfi/dchf-contracts/src/master/

...View All

COMMITS

- 409d3ea304cf130bff6f2f5d9a3ee4881972fe48
- 901c1b05372fbc17bc3474152e9a3916a119d96a

...View All

Vulnerability Summary

24	18	2	2	2	0	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined	Unresolved

0	Critical		Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
2	Major	2 Mitigated	Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
5	Medium	5 Resolved	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
12	Minor	9 Resolved, 1 Partially Resolved, 2 Acknowledged	Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
5	Informational	4 Resolved, 1 Partially Resolved	Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | DEFI FRANC

Summary

Executive Summary

Vulnerability Summary

Codebase

Audit Scope

Approach & Methods

Review Notes

Overview

External Dependencies
Privileged Functions

Findings

GLOBAL-01 : Centralization Related Risks

GLOBAL-02 : Lack of Storage Gap

CIM-01 : Potential Incorrect Issuance in `issueMON()`

CKP-01 : Lack of check on `adminContract`

CKP-02 : Unchecked ERC-20 `transfer()`/`transferFrom()` Call

DPK-01 : Lack of input validation

ERP-01 : Susceptible to Signature Malleability

HHC-01 : Potential underflow revert in `getRedemptionHints()`

LMO-01 : Divide Before Multiply

MOC-01 : Potential Reentrancy Attack (Incrementing State)

MOC-02 : Check Effect Interaction Pattern Violated

MOT-01 : Initial token distribution

STC-01 : Incorrect input used

STC-02 : Lack of input validation

STD-01 : Incompatible with tokens with more than 18 decimals

TMC-01 : Uncallable function in `TroveManager`

TMH-01 : Incorrect modifier

TMH-02 : Uncallable functions in `TroveManagerHelpers`

TMH-03 : Lack of input validation

CKP-04 : Redundant Code Components

CKP-05 : Missing Error Messages

CKP-06 : Missing Emit Events

CKP-07 : Missing Zero Address Validation

TMH-04 : Repetitive function implementation

Optimizations

GLOBAL-03 : Unnecessary Use of SafeMath and SafeMathUpgradeable

BOC-01 : Useless Statement

CKP-03 : Improper Usage of `public` and `external` Type

DMD-01 : Unnecessary write to memory

LMO-02 : Costly Operation Inside Loop

MOT-02 : State Variable Should Be Declared Constant

TMC-02 : Unnecessary external call

Appendix

Disclaimer

CODEBASE | DEFI FRANC

Repository

https://bitbucket.org/grizzlyfi/dchf-contracts/src/master/

Commit

- 409d3ea304cf130bff6f2f5d9a3ee4881972fe48
- 901c1b05372fbc17bc3474152e9a3916a119d96a

AUDIT SCOPE | DEFI FRANC

46 files audited · 2 files with Acknowledged findings · 10 files with Partially Resolved findings · 1 file with Mitigated findings · 2 files with Resolved findings · 31 files without findings

ID	File	SHA256 Checksum
ERP	Dependencies/ERC20Permit.sol	52294eb37c593c2f8aac7a6d9f9b968f2fe96c0f7d5008500b0664ce729b2efa
HHC	HintHelpers.sol	cdd5abedbf395703a98cffb121f409d497977b23c35b65f37f744cb336b85d1c
CIM	MON/CommunityIssuance.sol	f27be3533c0718d70dbdf4cf387b97860b3ab387f5abd4ff6aab0521fe27e385
LMO	MON/LockedMON.sol	31b1dd252ac3a73df0d0361aebaedbaaa4af0a1d053a818a794c789a9f6b193b
MOS	MON/MONStaking.sol	e83154ccd4f718fc3699fe7f7b35e0831da0e5eff311bf00f67d4427f663cf55
MOT	MON/MONToken.sol	5eca88e804dc270c9f7a58d0acd4cce651a9e171961b7fe03857ffb36458eed7
ACC	AdminContract.sol	6327f21d4638095f60a1215e336cd6541c8b8df6e0b639d5b8ecc7c14faa2869
BOC	BorrowerOperations.sol	9dfc8e397151dd5723023b05750930226e2137583fb147757c8f1dcb871d7294
CSP	CollSurplusPool.sol	a648c30bb0258cc0a4c503381282cd5f48f665f1341783c21af8a4863c731b0d
DCH	DCHFToken.sol	844c309c305c02b3cf87a32af0ff8b4372f4a0f290e90c05198bbb7333640ecd
DPK	DfrancParameters.sol	c54b111347f4590aa10210f59ede1f873da0da457b5f6c8ac65d23a41765a4cd
MCK	Migrations.sol	6f5d4f27d32f59aaf2a1b2b0130022f151e925522c406152bd1cbb5826d711fa
PFC	PriceFeed.sol	5f12482b27994c13daf9909bcc34454f63dd65b727066b4ac1ea6f3c241716b0
DMD	Dependencies/DfrancMath.sol	1abb322c263aeb9f815495b2628b030d04688e71db5dee62ee406cf10193b8ed

ID	File	SHA256 Checksum
STD	Dependencies/SafetyTransfer.sol	9d7b0d104ce49c2e923920463f3c043aaa28d6c5f9794ff661d9b0ae5187ecd1
BMD	Dependencies/BaseMath.sol	4421956cd5b4684bff063f63e9b01433f7d7d4f92d398f405ac380715da5d35a
CCD	Dependencies/CheckContract.sol	fb24cbcfaf9c19cf7b26e940986fdd0c6abf984c7152ce55e911640d2a8310fb
DBD	Dependencies/DfrancBase.sol	e99b1c0d59a7c2905fc8a45e71a5f48f30fbd5158fdf37a53f18f4ac2a56c150
DSM	Dependencies/DfrancSafeMath128.sol	d1bcf6981f794fb07c4dbec19d661fadb7d614942faa39b1c334b8f846ea363f
ERC	Dependencies/ERC20Decimals.sol	6061d0e907906f9bc0e7d801c2609f8d66ffc1bca5413c2fee8e1501b0df1d1e
IER	Dependencies/IERC2612.sol	d50c8ca19df49c1c487d5aba4513cbc9f849844eea51726c5db1c2f30ff0fcda
ITD	Dependencies/ITellor.sol	e868f248be4b7459fe85e2421411eeb416c58ca68153cfa0d9ed4735a17e146a
TCD	Dependencies/TellorCaller.sol	d65bfbdf958d1e0b688e50cd3ba88f314e9345f9ab3a08ab8edfc10e32a27e6d
IAP	Interfaces/IActivePool.sol	ae72a271ce99f4dbd14cd61ce57f40af85e24398bacc82a5921407ff9278f5ce
IBO	Interfaces/IBorrowerOperations.sol	f3b37a3718c685dbb01c7db1d8e5367be93f169d6a55a412ad12bad87eedd932
ICS	Interfaces/ICollSurplusPool.sol	30e5040f66bc3c166b1cfd8ab2646295aa8d95da1edb4278db56b2f1725b4ab8
ICI	Interfaces/ICommunityIssuance.sol	3045a4a188b4961ea58a1088cec8eaafdc5abc6e1f3dba73038d0ecec6a784a3
IDC	Interfaces/IDCHFToken.sol	8239d56a53e06446717d0ba50a4bfb20065320bda3441051c73d4f061a65346c
IDP	Interfaces/IDefaultPool.sol	39f533af5c3dfeb37662852e24daf16d35a1dbb83ba2675392aa44d4042f1734
IDI	Interfaces/IDeposit.sol	d687f9d7a4a5a84b2eade4e00baa78a382ace8547ff38ae27a634210f503d463

ID	File	SHA256 Checksum
IDB	Interfaces/IDfrancBase.sol	6f9f332c0e018fb42db6187a4224b673da984d9dad9a4e084b677194f5ebf9ae
IDK	Interfaces/IDfrancParameters.sol	8f8f07ccab997e841ee225aafabb3e57a17313ffc6bdfee07daa49d1e931e1f1
IEC	Interfaces/IERC20Deposit.sol	85ef7973bc8566876a7c81e81203b0405fd82c88e27d2a1124aae8c64d524d88
IMO	Interfaces/IMONStaking.sol	933260dc60f8099774dfc562819cfa5b6c16ed5b5d3b21a8df5da96cbb731e32
IPI	Interfaces/IPool.sol	f7373cd15ac184b3d47968e04cda7ed575627c237fdd601412fd180358b09670
IPF	Interfaces/IPriceFeed.sol	ca49db882883354e02ab9939dd5c8633047e3c701014b6c0744b9211383a245f
IST	Interfaces/ISortedTroves.sol	b2ff32b9b72a9b16c33f58f85c4d1ebd5c234f716f6e9545eddd7d0ecbdad445
ISP	Interfaces/IStabilityPool.sol	f36d41b17b3f2198bbbba14520adba4f86ecb398375146e5580e235b4a738200
ISM	Interfaces/IStabilityPoolManager.sol	cbb77ad06363237bc366ffedba6237b810f2ab07134c63869566de15345fc490
ITC	Interfaces/ITellorCaller.sol	58744f4e70b38d0f58a3a78f5f246254c88ed433c4bd4c6e29a4120395f16875
ITM	Interfaces/ITroveManager.sol	92c7eb950740c85ab9b58019c45b685370e46c9dc9c8f1338a6abda4143ac688
ITH	Interfaces/ITroveManagerHelpers.sol	b7de30ebe4a4bd08a7c1427507274de5c14b5b3f045d75bb892cc4808ea333b2
APC	ActivePool.sol	6d32a95841a6345cae21d99c3b8f1b7c194c9d0451b7db789e53ce2a953d5e55
DPC	DefaultPool.sol	3aca1b24191cf54b277fd8fbed803c5eb5b5d9348ca8775df74ccd022e3c8615
GPC	GasPool.sol	d9496c8d3054b7f85f36d455e4539f719d43a41cf4b8687cf234943d04a1098f
MTG	MultiTroveGetter.sol	a3b52c6ddf33e91eb74ec9e00d828f011e95f12eda525e9a94e0adbbef5cf56f

APPROACH & METHODS | DEFI FRANC

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

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

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

- Testing the smart contracts against both common and uncommon attack vectors;
- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

REVIEW NOTES | DEFI FRANC

Overview

DeFi Franc is a liquidity mining aggregator that aggregates liquidity mining opportunities throughout the ecosystem.

The smart contracts in the scope of the audit were forked from Liquity and Vesta Finance, which are protocols for collateralized liquidity mining. It allows users to use the native token as collateral in order to borrow the CHF stablecoin DCHF, with zero percent interest. Furthermore, users can stake their stablecoin to earn the reward token Moneta or use the stablecoin to redeem the native token at face value, regardless of the price of the stablecoin.

External Dependencies

The scope of the audit treats third-party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can be compromised and this may lead to lost or stolen assets.

There are a few dependent injection contracts or addresses in the current project:

- Chainlink oracle;
- Collateral assets.

Privileged Functions

In the `DeFi Franc` project, multiple privileged roles are adopted to ensure the dynamic runtime updates of the project, which were specified in the finding *GLOBAL-01 | Centralization Related Risks*.

The advantage of those privileged roles in the codebase is that the client reserves the ability to adjust the protocol according to the runtime required to best serve the community. It is also worth noting the potential drawbacks of these functions, which should be clearly stated through the client's action/plan. Additionally, if the private keys of the privileged accounts are compromised, it could lead to a devastating consequence to the project.

To improve the trustworthiness of the project, dynamic runtime updates in the project should be notified to the community. Any plan to invoke the aforementioned functions should be also considered to move to the execution queue of the `Timelock` contract.

FINDINGS | DEFI FRANC

24	0	2	5	12	5
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for DeFi Franc. Through this audit, we have uncovered 24 issues ranging from different severity levels. Utilizing Static Analysis techniques to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
GLOBAL-01	**Centralization Related Risks**	**Centralization / Privilege**	**Major**	**Mitigated**
GLOBAL-02	Lack Of Storage Gap	Language Specific	Medium	Resolved
CIM-01	Potential Incorrect Issuance In `issueMON()`	Logical Issue	Minor	Resolved
CKP-01	Lack Of Check On `adminContract`	Inconsistency	Minor	Resolved
CKP-02	Unchecked ERC-20 `transfer()` / `transferFrom()` Call	Volatile Code	Minor	Resolved
DPK-01	Lack Of Input Validation	Volatile Code	Minor	Partially Resolved
ERP-01	Susceptible To Signature Malleability	Volatile Code	Minor	Acknowledged
HHC-01	Potential Underflow Revert In `getRedemptionHints()`	Logical Issue	Minor	Acknowledged
LMO-01	Divide Before Multiply	Mathematical Operations	Minor	Resolved
MOC-01	Potential Reentrancy Attack (Incrementing State)	Volatile Code	Minor	Resolved

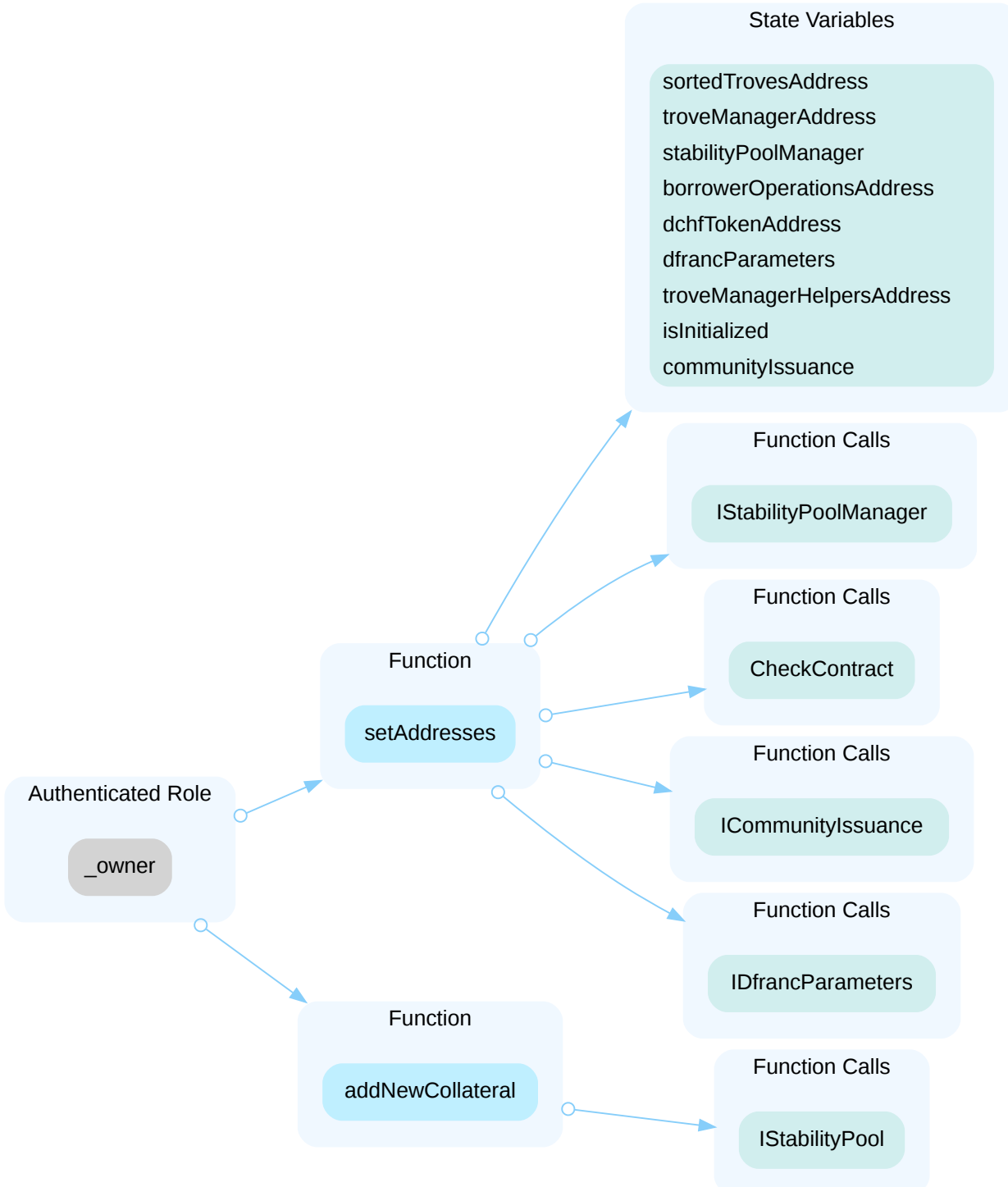
ID	Title	Category	Severity	Status
MOC-02	Check Effect Interaction Pattern Violated	Volatile Code	Minor	Resolved
MOT-01	**Initial Token Distribution**	**Centralization / Privilege**	**Major**	**Mitigated**
STC-01	Incorrect Input Used	Volatile Code	Medium	Resolved
STC-02	Lack Of Input Validation	Inconsistency	Minor	Resolved
STD-01	Incompatible With Tokens With More Than 18 Decimals	Control Flow	Minor	Resolved
TMC-01	Uncallable Function In `TroveManager`	Volatile Code	Medium	Resolved
TMH-01	Incorrect Modifier	Inconsistency	Medium	Resolved
TMH-02	Uncallable Functions In `TroveManagerHelpers`	Volatile Code	Medium	Resolved
TMH-03	Lack Of Input Validation	Inconsistency	Minor	Resolved
CKP-04	Redundant Code Components	Volatile Code	Informational	Resolved
CKP-05	Missing Error Messages	Coding Style	Informational	Resolved
CKP-06	Missing Emit Events	Coding Style	Informational	Partially Resolved
CKP-07	Missing Zero Address Validation	Volatile Code	Informational	Resolved
TMH-04	Repetitive Function Implementation	Coding Style	Informational	Resolved

GLOBAL-01 | CENTRALIZATION RELATED RISKS

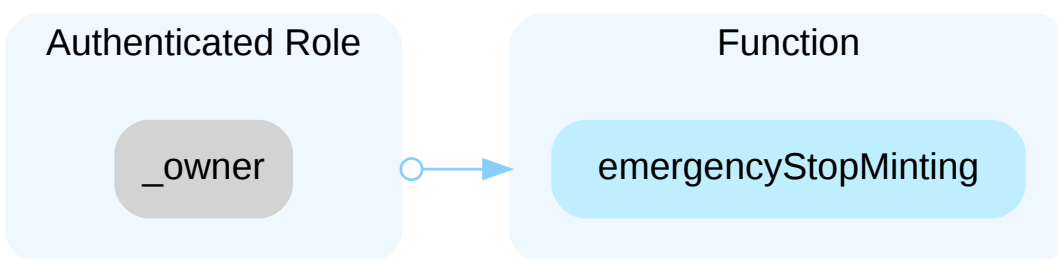
Category	Severity	Location	Status
Centralization / Privilege	**Major**		**Mitigated**

Description

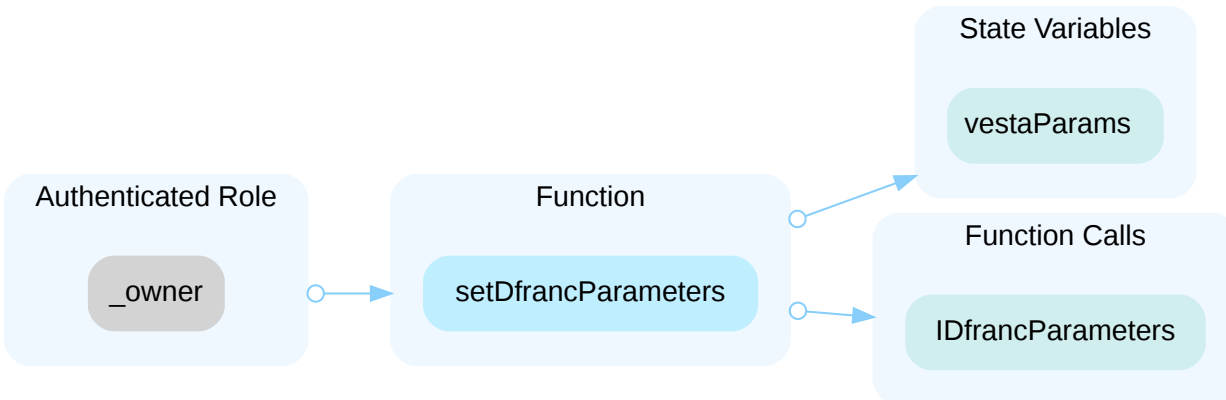
In the contract `AdminContract` , the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and add his own custom tokens as collateral.



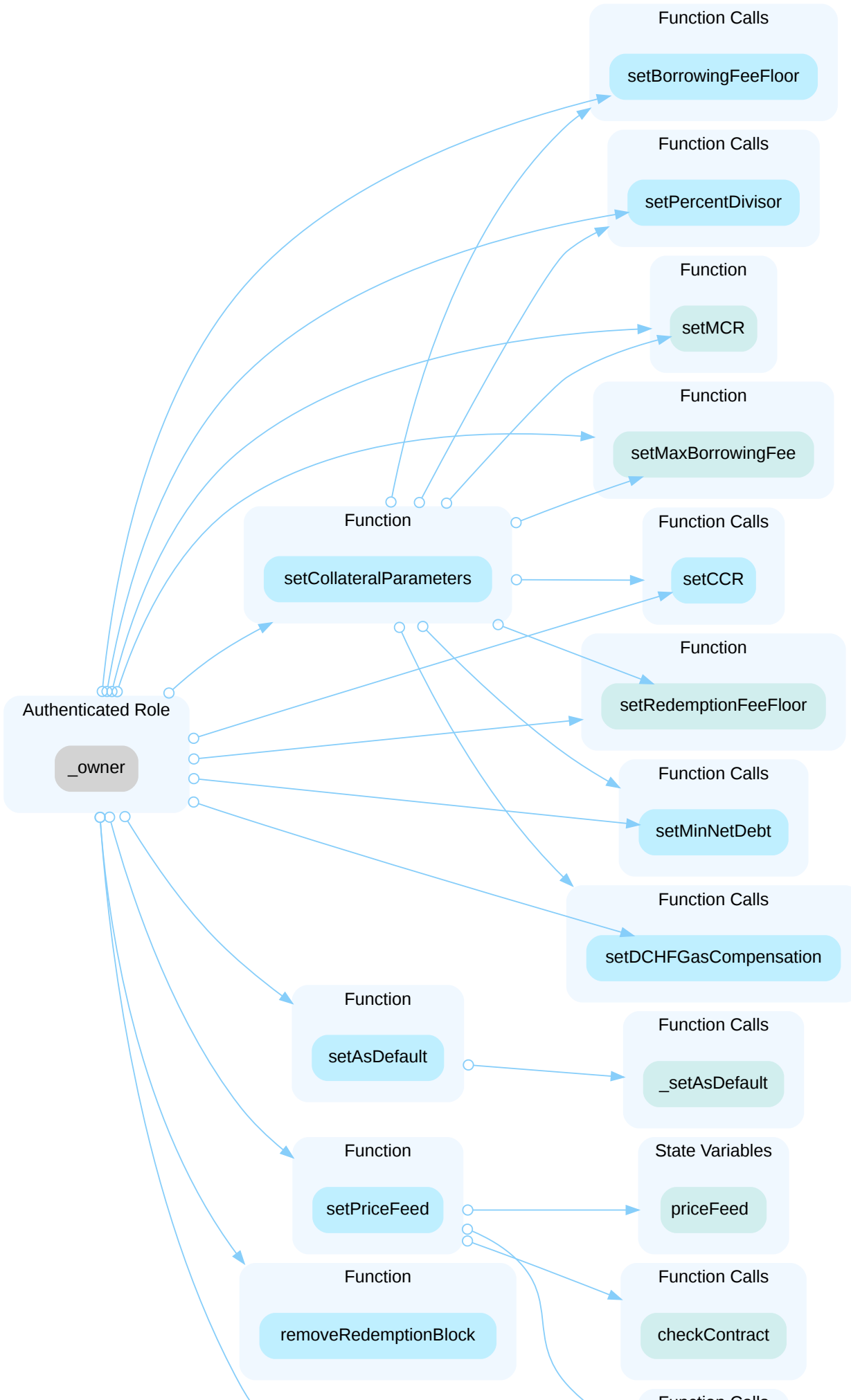
In the contract `DCHFToken`, the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and stop the minting of `DCHF`.

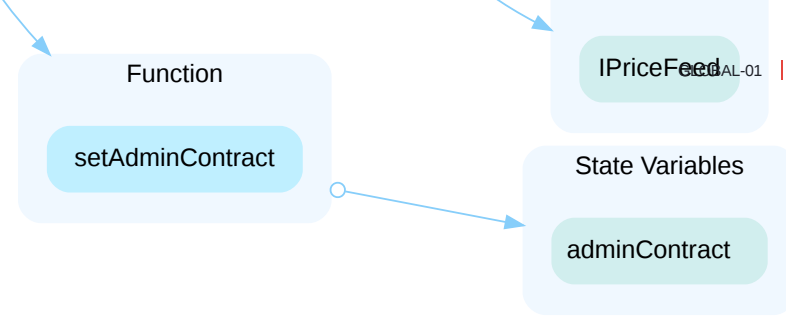


In the contract `DfrancBase`, the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and modify the address for protocol parameters.

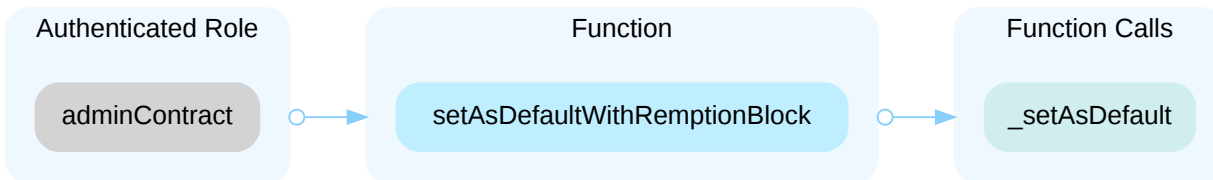


In the contract `DfrancParameters`, the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and modify the admin contract or change protocol parameters.

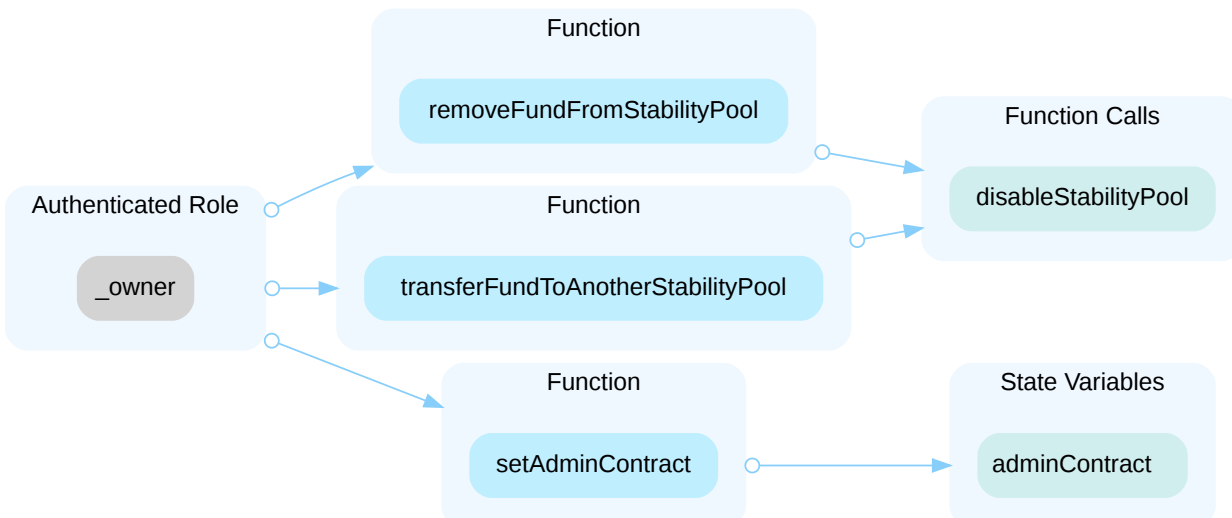

Function Calls
setBorrowingFeeFloor
Function Calls
setPercentDivisor
Function
setMCR
Function
setMaxBorrowingFee
Function
setCollateralParameters
Function Calls
setCCR
Function
setRedemptionFeeFloor
Authenticated Role
_owner
Function Calls
setMinNetDebt
Function Calls
setDCHFGasCompensation
Function
setAsDefault
Function Calls
_setAsDefault
Function
setPriceFeed
State Variables
priceFeed
Function
removeRedemptionBlock
Function Calls
checkContract



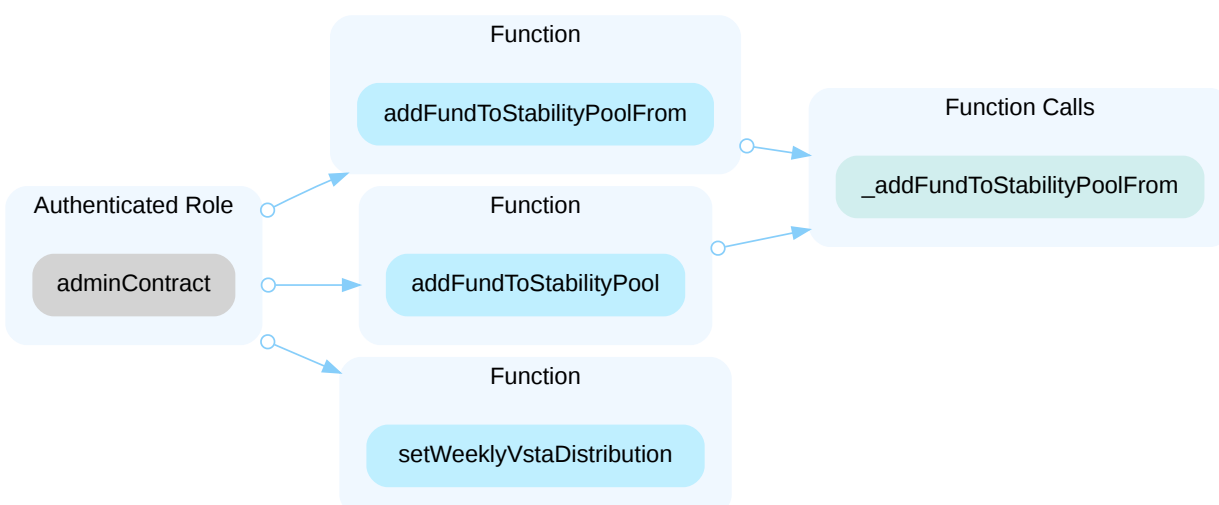
In the contract `DfrancParameters`, the role `adminContract` has authority over the functions shown in the diagram below. Any compromise to the `adminContract` account may allow the hacker to take advantage of this authority and configure some state variables for an asset.



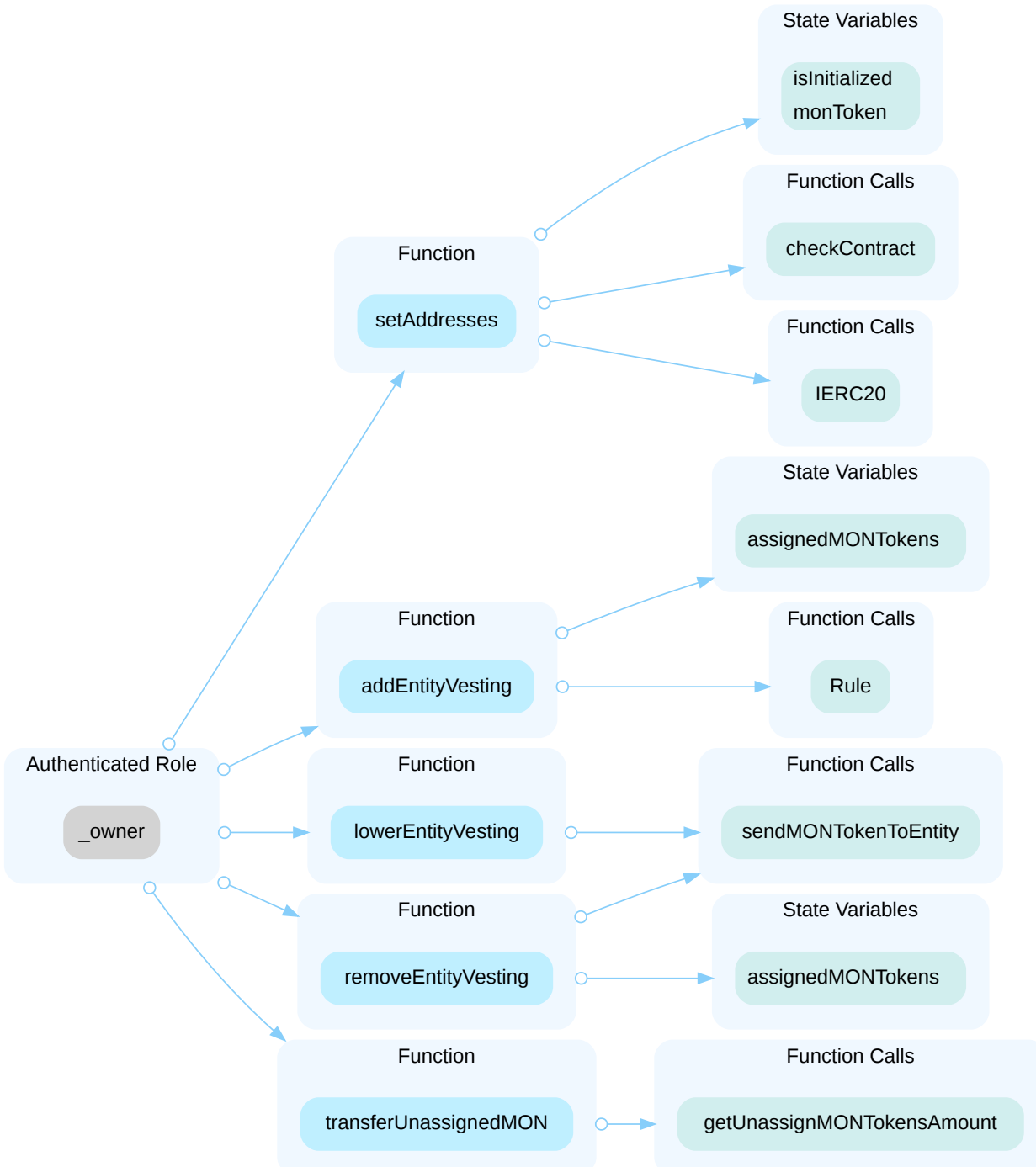
In the contract `CommunityIssuance`, the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and modify the admin contract.



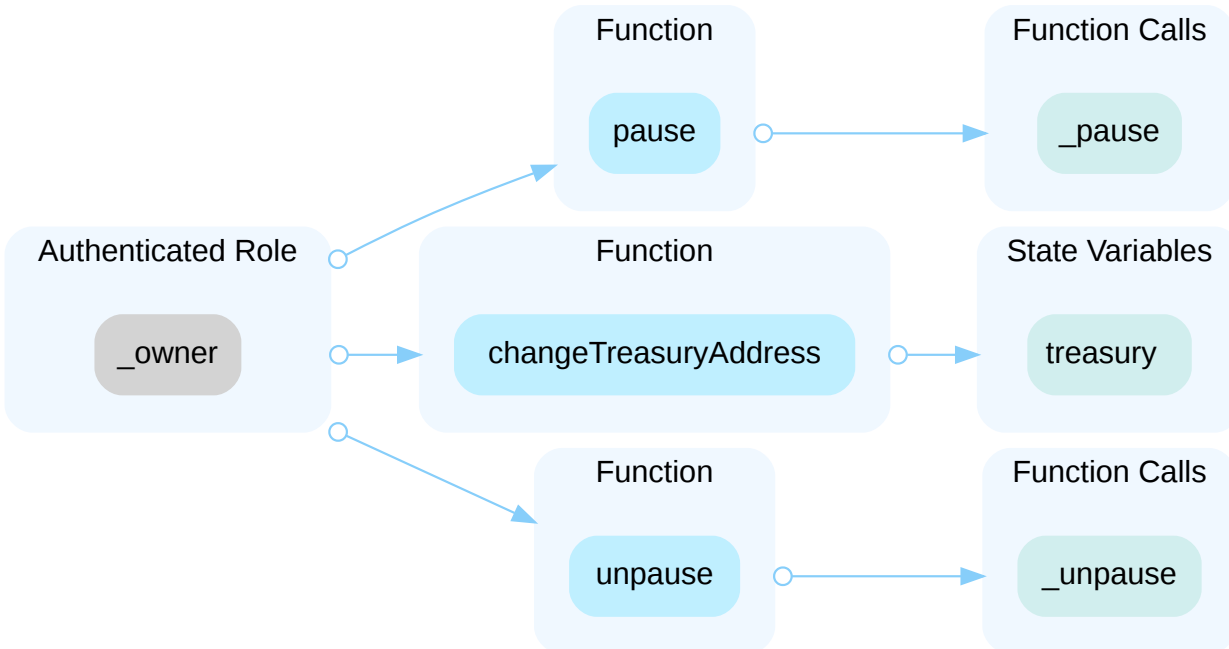
In the contract `CommunityIssuance`, the role `adminContract` has authority over the functions shown in the diagram below. Any compromise to the `adminContract` account may allow the hacker to take advantage of this authority and modify the weekly distribution of tokens.



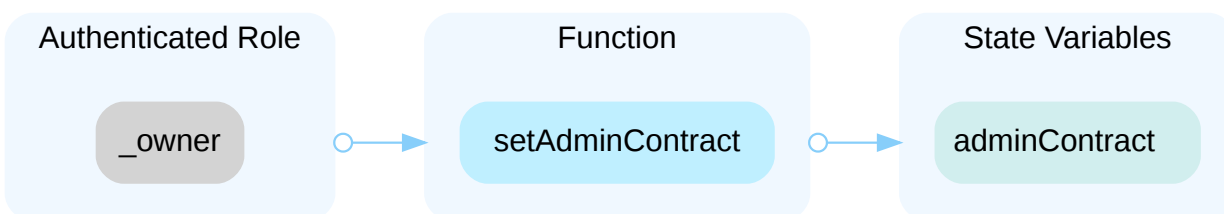
In the contract `LockedMON`, the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and remove an entity from the `entitiesVesting` array.



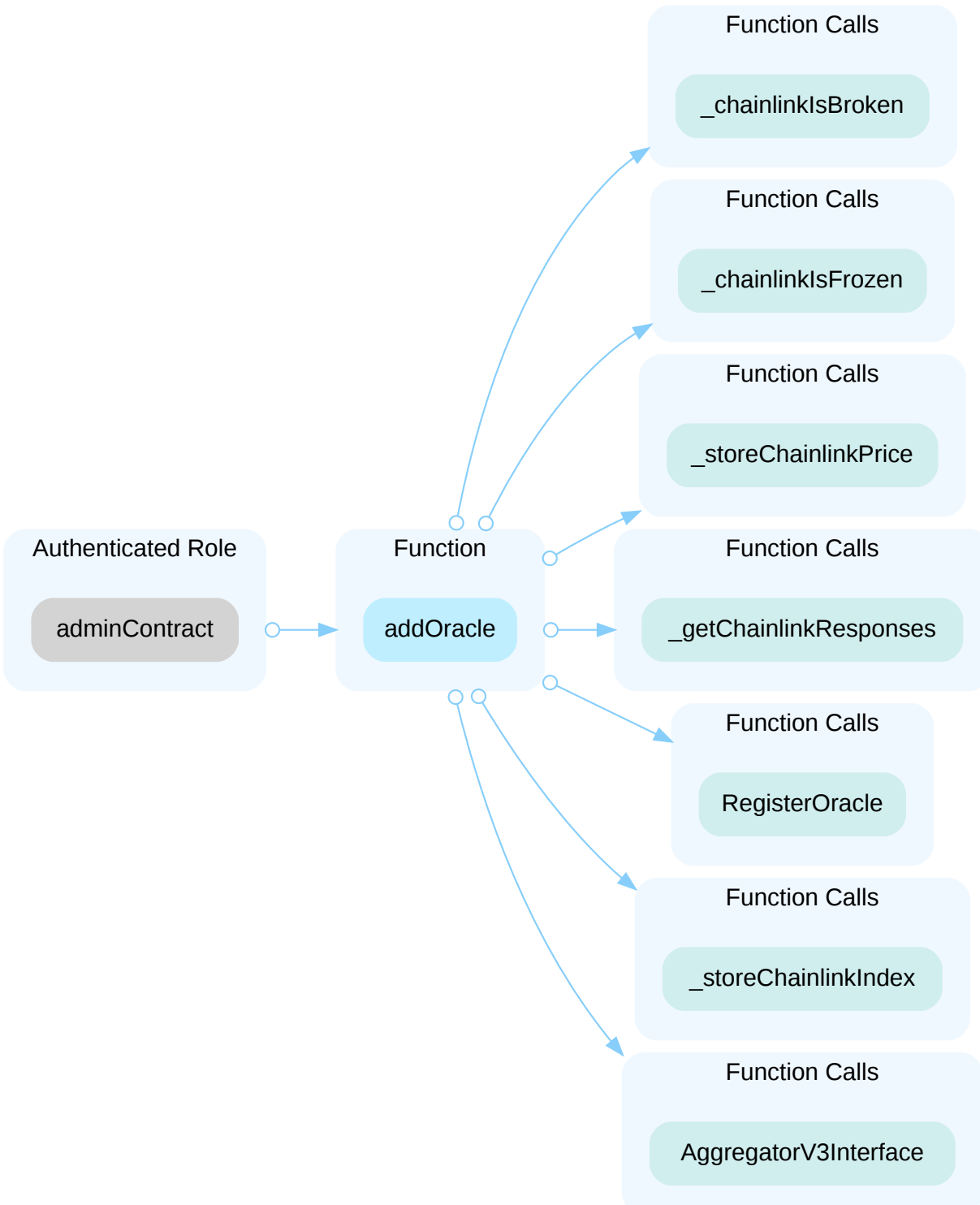
In the contract `MONStaking`, the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and pause the contract.



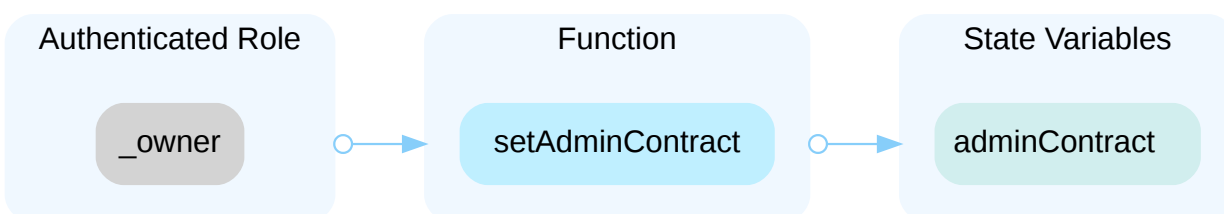
In the contract `PriceFeed`, the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and modify the admin contract.



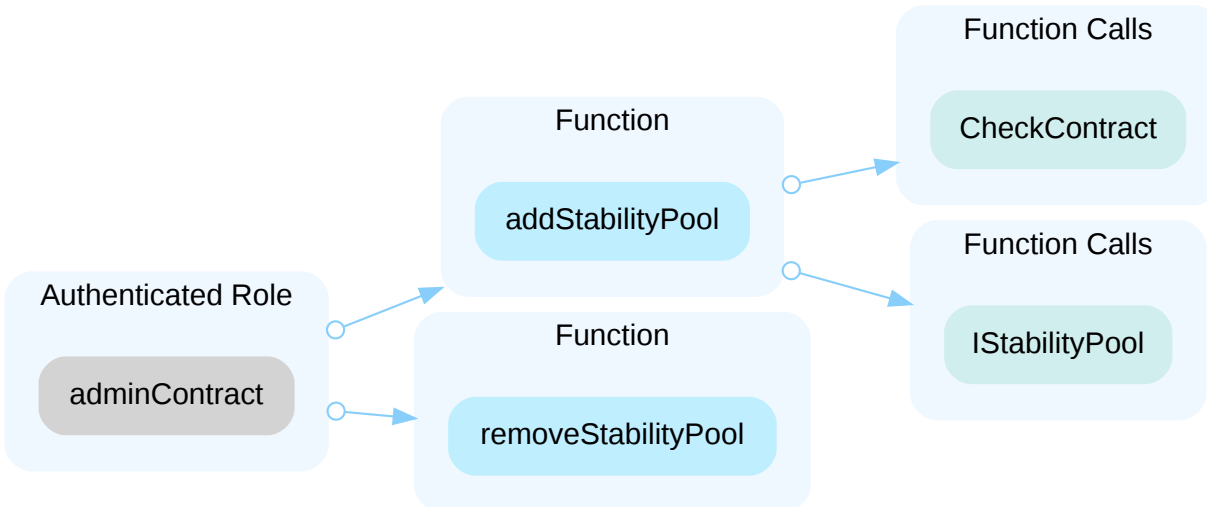
In the contract `PriceFeed`, the role `adminContract` has authority over the functions shown in the diagram below. Any compromise to the `adminContract` account may allow the hacker to take advantage of this authority and add Chainlink oracles.



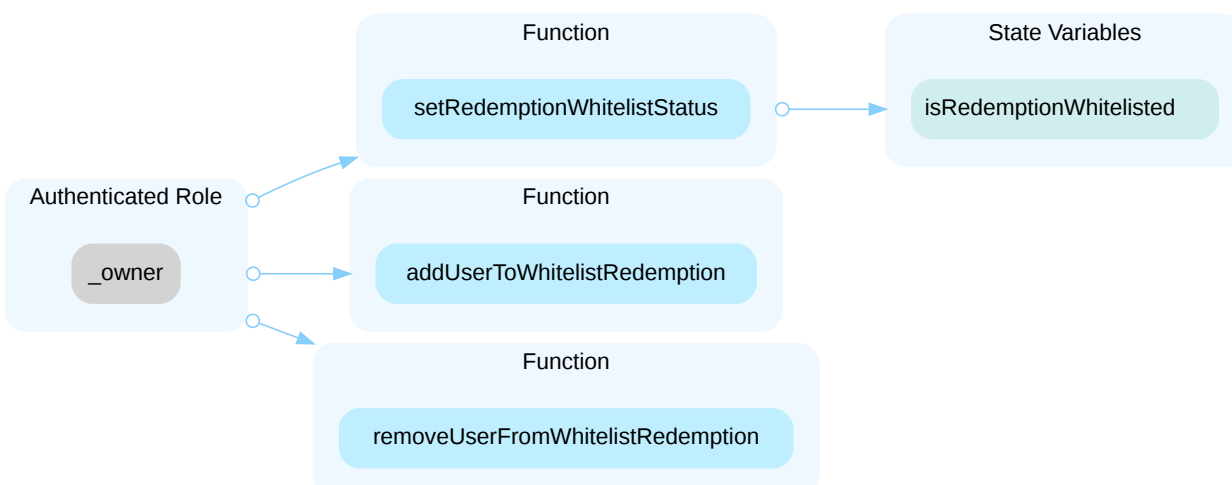
In the contract `StabilityPoolManager`, the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and modify the admin contract.



In the contract `StabilityPoolManager`, the role `adminContract` has authority over the functions shown in the diagram below. Any compromise to the `adminContract` account may allow the hacker to take advantage of this authority and add or remove stability pools.



In the contract `TroveManager`, the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and add or remove users from the `WhitelistRedemption` array.



In addition, the contracts

- `ActivePool`,
- `AdminContract`,
- `BorrowerOperations`,
- `CollSurplusPool`,
- `DefaultPool`,
- `DfrancBase`,
- `DfrancParameters`,
- `HintHelpers`,

- `PriceFeed`,
- `SortedTroves`,
- `StabilityPool`,
- `StabilityPoolManager`,
- `TroveManager`,
- `TroveManagerHelpers`,
- `CommunityIssuance`,
- `LockedMON`,
- `MONStaking`,
- `PriceFeed`,
- `SortedTroves`,
- `StabilityPoolManager`

are upgradeable contracts, meaning the owner can upgrade the contract without the community's consensus. If an attacker compromises the account, they can change the implementation of the contract and drain tokens from the contract.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

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

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement;
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles;
  OR
- Remove the risky functionality.

Noted: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

[DeFi Franc] :
The team has implemented the following short term solution:

1. Multi-sign proxy address:
   - https://etherscan.io/address/0x83737eae72ba7597b36494d723fbf58cafee8a69
2. Transaction proof for transferring ownership to multi-signature proxy:
   - https://etherscan.io/tx/0xea7d8303eb36885d2446bd3ea73ca64027f5e851c5ba0119fddd0370b3604468
3. Internal multi-signature address:
   - https://etherscan.io/address/0x8c013078c75e790Ffed8E11342EcfF53c5cd73A8,
   - https://etherscan.io/address/0x7AFF0f97357a7e8b577298f2fe81E6330975e28d,
   - https://etherscan.io/address/0x67733CFa01B42900057759a8EBA97AFED02C44E8

GLOBAL-02 | LACK OF STORAGE GAP

Category	Severity	Location	Status
Language Specific	Medium		Resolved

Description

`ActivePool`, `AdminContract`, `BorrowerOperations`, `CollSurplusPool`, `DefaultPool`, `DfrancBase`, `DfrancParameters`, `HintHelpers`, `PriceFeed`, `SortedTroves`, `StabilityPool`, `StabilityPoolManager`, `TroveManager`, `TroveManagerHelpers`, `CommunityIssuance`, `LockedMON`, `MONStaking`, `PriceFeed`, `SortedTroves`, `StabilityPoolManager` are upgradeable contracts.

For upgradeable contracts, there must be a storage gap to "allow developers to freely add new state variables in the future without compromising the storage compatibility with existing deployments". Otherwise, it may be very difficult to write new implementation code. Without a storage gap, the variable in a child contract might be overwritten by the upgraded base contract if new variables are added to the base contract.

Refer to https://docs.openzeppelin.com/upgrades-plugins/1.x/writing-upgradeable

Recommendation

It is recommended to add an appropriate storage gap at the end of each upgradeable contract.

Alleviation

[DeFi Franc] :

The team resolved the finding in commit d6f731b32d6b04aa65f987c330e2cdb108f28c54 by modifying the design and opting for non-upgradeable contracts.

CIM-01 | POTENTIAL INCORRECT ISSUANCE IN `issueMON()`

Category	Severity	Location	Status
Logical Issue	Minor	MON/CommunityIssuance.sol: 173~195	Resolved

Description

The function `issueMON()` is called by the `StabilityPool` and returns the new MON issuance of the pool. The value of the new issuance is the multiplication of the number of minutes from the `lastUpdateTime[pool]` to the `block.timestamp`, and the `monDistributionsByPool[stabilityPool]`.

The variable `lastUpdateTime[pool]` will update to `block.timestamp` in every call, unless the total issuance has met the cap.

Due to the division truncation in solidity, the `timePassed` will be 0 if the interval is less than one minute and the `issuance` will also be 0.

```
        uint256 timePassed = block.timestamp.sub(lastUpdateTime[stabilityPool]).div(
            SECONDS_IN_ONE_MINUTE
        );
        uint256 totalDistribuedSinceBeginning =
monDistributionsByPool[stabilityPool].mul(
            timePassed
        );
```

As a result, in the case of multiple calls where the interval between each call is less than one minute, the `issuance` returned is always 0 and the total issuance will not match the actual value.

Recommendation

The auditing team recommends updating `lastUpdateTime[_pool]` in terms of minutes instead of seconds.

Alleviation

[DeFi Franc] :

The team heeded the advice and resolved the finding in the commit hash f14dc00a17d3ec14e493648913c23533732934f0.

CKP-01 | LACK OF CHECK ON `adminContract`

Category	Severity	Location	Status
Inconsistency	Minor	DfrancParameters.sol: 76~79; MON/CommunityIssuance.sol: 83~86; PriceFeed.sol: 50~53; StabilityPoolManager.sol: 32~35	Resolved

Description

In the aforementioned contracts, upon initialization (`setAddresses()`), `adminContract` is checked to ensure that the address corresponds to a contract.
This verification is performed with the `checkContract()` function.
However, the owner can later change the address for an EOA, because the `checkContract()` verification is missing in the `setAdminContract()` function.

```
22      function setAddresses(address _adminContract) external initializer {
23          require(!isInitialized, "Already initialized");
24          checkContract(_adminContract);
25          isInitialized = true;
26
27          __Ownable_init();
28
29          adminContract = _adminContract;
30      }
31
32      function setAdminContract(address _admin) external onlyOwner {
33          require(_admin != address(0), "Admin cannot be empty address");
34          adminContract = _admin;
35      }
```

Recommendation

It is recommended to add the `checkContract()` verification inside the `setAdminContract()` function.

Alleviation

[DeFi Franc] :
The team heeded the advice and resolved the finding in the commit hash ae33c3a00044a91e8a8732d83639d27852fa8132.

CKP-02 | UNCHECKED ERC-20 `transfer()` / `transferFrom()` CALL

Category	Severity	Location	Status
Volatile Code	Minor	MON/CommunityIssuance.sol: 225; MON/MONStaking.sol: 118, 138, 161, 182; Proxy/TokenScript.sol: 19, 35	Resolved

Description

The return value of the transfer()/transferFrom() call is not checked.

```
225         monToken.transfer(_account, safeAmount);
```

```
118                     dchfToken.transfer(msg.sender, DCHFGain);
```

```
138         monToken.transferFrom(msg.sender, address(this), _MONamount);
```

```
161                 dchfToken.transfer(msg.sender, DCHFGain);
```

```
182             monToken.transfer(msg.sender, MONToWithdraw);
```

```
19          token.transfer(recipient, amount);
```

```
35          token.transferFrom(sender, recipient, amount);
```

Recommendation

Since some ERC-20 tokens return no values and others return a `bool` value, they should be handled with care. We advise using the OpenZeppelin's `SafeERC20.sol` implementation to interact with the `transfer()` and `transferFrom()` functions of external ERC-20 tokens. The OpenZeppelin implementation checks for the existence of a return value and reverts if `false` is returned, making it compatible with all ERC-20 token implementations.

Alleviation

[DeFi Franc] :

The team heeded the advice and resolved the finding in commit f6af9db8addb65a8dc181b6241970242d8cc21f5.

DPK-01 | LACK OF INPUT VALIDATION

Category	Severity	Location	Status
Volatile Code	Minor	DfrancParameters.sol: 149~150	Partially Resolved

Description

The following functions lack input validation and could intentionally or unintentionally break the protocol.

- The functions `setMCR()` and `setCCR()` can change CCR and MCR, which could result in CCR < MCR.
- The functions `setBorrowingFeeFloor()` and `setMaxBorrowingFee()` can change `MAX_BORROWING_FEE` and `BORROWING_FEE_FLOOR`, which could result in `MAX_BORROWING_FEE` < `BORROWING_FEE_FLOOR`.
- The function `setDCHFGasCompensation()` can increase `DCHF_GAS_COMPENSATION` which could result in not having enough DCHF tokens to burn from the gas pool, preventing liquidations and redemptions.

Recommendation

Consider adding input verification such as `require` statements to check if the new value is consistent with the system protocol and does not cause an error for operations on existing troves. The following are some possible solutions.

- For the functions `setMCR()` and `setCCR()`, include `require(MCR[_asset] < CCR[_asset])` after the new value is set.
- For the functions `setBorrowingFeeFloor()` and `setMaxBorrowingFee()`, include `require(MAX_BORROWING_FEE[_asset] > BORROWING_FEE_FLOOR[_asset])`
- For the function `setDCHFGasCompensation()`, mint or burn to the gas pool whenever `DCHF_GAS_COMPENSATION` is changed to ensure enough tokens are available in the gas pool.

Alleviation

[DeFi Franc]

The team heeded the advice and partially resolved the finding in the commit hash 3df6e56e38aa045c9abdad48a796b916b6f76edd by adding checks on the borrowing fee and only allowing the first change of the gas compensation to be a decrease. However, it is still possible for the MCR of an asset to be above the CCR.

ERP-01 | SUSCEPTIBLE TO SIGNATURE MALLEABILITY

Category	Severity	Location	Status
Volatile Code	Minor	Dependencies/ERC20Permit.sol: 99	Acknowledged

Description

The signature malleability is possible within the Elliptic Curve cryptographic system. An Elliptic Curve is symmetric on the X-axis, meaning two points can exist with the same `X` value. In the `r`, `s` and `v` representation this permits us to carefully adjust `s` to produce a second valid signature for the same `r`, thus breaking the assumption that a signature cannot be replayed in what is known as a replay-attack.

Recommendation

We advise to utilize a `recover()` function similar to that of the `ECDSA.sol` implementation of OpenZeppelin.

Alleviation

[DeFi Franc] :
The team acknowledged the advice and will not change the current codebase.

HHC-01 POTENTIAL UNDERFLOW REVERT IN `getRedemptionHints()`

Category	Severity	Location	Status
Logical Issue	Minor	HintHelpers.sol: 118	Acknowledged

Description

According to the following code, the variable `_maxIterations` is used as an iteration condition for the while loop and is self-subtracting after each iteration.

```
        if (_maxIterations == 0) {
            _maxIterations = type(uint256).max;
        }

        while (currentTroveuser != address(0) && remainingDCHF > 0 &&
_maxIterations-- > 0) {
           ...
        }
```

If in the loop, the variable `_maxIterations` reaches 1 and the loop has not stopped, the variable `_maxIterations` (1) will be compared to 0 and subtracted. In the next iteration, the `_maxIterations` is 0 and the iteration is false. However, the operation `--` will be done and it will trigger an underflow revert, since the compiler version is `^0.8.14`.

Recommendation

The auditing team recommends placing the `--` operation within the while loop.

Alleviation

[DeFi Franc]

The team acknowledged the finding and decided to keep the codebase unchanged.

LMO-01 | DIVIDE BEFORE MULTIPLY

Category	Severity	Location	Status
Mathematical Operations	Minor	MON/LockedMON.sol: 157~161	Resolved

Description

Performing integer division before multiplication truncates the lower bits, losing the precision of calculation.

```
157             claimable = entityRule
158                 .totalSupply
159                 .div(ONE_YEAR)
160                 .mul(block.timestamp.sub(entityRule.createdDate))
161                 .sub(entityRule.claimed);
```

Recommendation

Apply multiplication before division to avoid loss of precision.

Alleviation

[DeFi Franc] :

The team heeded the advice and resolved the finding in the commit hash 9eff3050bd1021fd1896d66e6d6fe43931628557.

MOC-01 POTENTIAL REENTRANCY ATTACK (INCREMENTING STATE)

Category	Severity	Location	Status
Volatile Code	Minor	MON/LockedMON.sol: 103, 104, 111, 137; MON/MONStaking.sol: 245, 246, 304, 307	Resolved

Description

A reentrancy attack can occur when the contract creates a function that makes an external call to another untrusted contract before resolving any effects. If the attacker can control the untrusted contract, they can make a recursive call back to the original function, repeating interactions that would have otherwise not run after the external call resolved the effects.

This finding is considered minor because the state variable is only incremented or decremented. So, the effect of out-of-order increments may be unobservable after transaction. However, the reentrancy vulnerability may still cause other issues in the middle of transaction.

External call(s)

```
103         sendMONTokenToEntity(_entity);
```

- This function call executes the following external call(s).
- In `SafeERC20._callOptionalReturn`,
  - `returndata = address(token).functionCall(data,SafeERC20: low-level call failed)`
- In `Address.functionCallWithValue`,
  - `(success,returndata) = target.call{value: value}(data)`
- In `LockedMON.sendMONTokenToEntity`,
  - `monToken.safeTransfer(_entity,unclaimedAmount)`

State variables written after the call(s)

```
104         Rule storage vestingRule = entitiesVesting[_entity];
```

```
111         vestingRule.totalSupply = newTotalSupply;
```

External call(s)

```
245         _sendAsset(treasury, _asset, _amount);
```

- This function call executes the following external call(s).
- In `SafeERC20Upgradeable._callOptionalReturn`,
  - `returndata = address(token).functionCall(data,SafeERC20: low-level call failed)`
- In `MONStaking._sendAsset`,
  - `(success) = _sendTo.call{value: _amount}()`
- In `AddressUpgradeable.functionCallWithValue`,
  - `(success,returndata) = target.call{value: value}(data)`
- In `MONStaking._sendAsset`,
  - `IERC20Upgradeable(_asset).safeTransfer(_sendTo,_amount)`

State variables written after the call(s)

```
246         sentToTreasuryTracker[_asset] += _amount;
```

Recommendation

We recommend using the Checks-Effects-Interactions Pattern to avoid the risk of calling unknown contracts or applying OpenZeppelin ReentrancyGuard library - `nonReentrant` modifier for the aforementioned functions to prevent reentrancy attack.

Alleviation

[DeFi Franc] :
The team heeded the advice and resolved this issue by adding the `nonReentrant` modifier, in the commit e3492f4df18d711fd8962cbeefdf4371f645ec92.

MOC-02 | CHECK EFFECT INTERACTION PATTERN VIOLATED

Category	Severity	Location	Status
Volatile Code	Minor	MON/LockedMON.sol: 115, 118~120, 122, 137; MON/MONStaking.sol: 118, 123, 127, 161, 165, 287, 288, 304, 307	Resolved

Description

A reentrancy attack can occur when the contract creates a function that makes an external call to another untrusted contract before resolving any effects. If the attacker can control the untrusted contract, they can make a recursive call back to the original function, repeating interactions that would have otherwise not run after the external call resolved the effects.

External call(s)

```
115         sendMONTokenToEntity(_entity);
```

- This function call executes the following external call(s).
- In `SafeERC20._callOptionalReturn`,
  - `returndata = address(token).functionCall(data,SafeERC20: low-level call failed)`
- In `Address.functionCallWithValue`,
  - `(success,returndata) = target.call{value: value}(data)`
- In `LockedMON.sendMONTokenToEntity`,
  - `monToken.safeTransfer(_entity,unclaimedAmount)`

State variables written after the call(s)

```
118         assignedMONTokens = assignedMONTokens.sub(
119             vestingRule.totalSupply.sub(vestingRule.claimed)
120         );
```

```
122         delete entitiesVesting[_entity];
```

External call(s)

```
118                     dchfToken.transfer(msg.sender, DCHFGain);
```

```
123                 _sendAssetGainToUser(asset, AssetGain);
```

- This function call executes the following external call(s).
- In `SafeERC20Upgradeable._callOptionalReturn`,
  - `returndata = address(token).functionCall(data,SafeERC20: low-level call failed)`
- In `MONStaking._sendAsset`,
  - `(success) = _sendTo.call{value: _amount}()`
- In `AddressUpgradeable.functionCallWithValue`,
  - `(success,returndata) = target.call{value: value}(data)`
- In `MONStaking._sendAsset`,
  - `IERC20Upgradeable(_asset).safeTransfer(_sendTo,_amount)`

State variables written after the call(s)

```
127             _updateUserSnapshots(asset, msg.sender);
```

- This function call executes the following assignment(s).
- In `MONStaking._updateUserSnapshots`,
  - `snapshots[_user].F_ASSET_Snapshot[_asset] = F_ASSETS[_asset]`
- In `MONStaking._updateUserSnapshots`,
  - `snapshots[_user].F_DCHF_Snapshot = F_DCHF`

External call(s)

```
161                 dchfToken.transfer(msg.sender, DCHFGain);
```

State variables written after the call(s)

```
165             _updateUserSnapshots(asset, msg.sender);
```

- This function call executes the following assignment(s).
- In `MONStaking._updateUserSnapshots`,
  - `snapshots[_user].F_ASSET_Snapshot[_asset] = F_ASSETS[_asset]`
- In `MONStaking._updateUserSnapshots`,
  - `snapshots[_user].F_DCHF_Snapshot = F_DCHF`

Recommendation

We recommend using the Checks-Effects-Interactions Pattern to avoid the risk of calling unknown contracts or applying OpenZeppelin ReentrancyGuard library - `nonReentrant` modifier for the aforementioned functions to prevent reentrancy attack.

Alleviation

[DeFi Franc] :
The team heeded the advice and resolved this issue by adding the `nonReentrant` modifier, in the commit e3492f4df18d711fd8962cbeefdf4371f645ec92.

MOT-01 | INITIAL TOKEN DISTRIBUTION

Category	Severity	Location	Status
Centralization / Privilege	**Major**	**MON/MONToken.sol: 20~22**	**Mitigated**

Description

All of the `MON` tokens are sent to the `_treasurySig` address when deploying the contract. This could be a centralization risk as this address can distribute MON tokens without obtaining the consensus of the community.

Recommendation

We recommend the team to be transparent regarding the initial token distribution process, and the team shall make enough efforts to restrict the access of the private key.

Alleviation

[DeFi Franc] :
The team published a Tokenomics document describing how the MON tokens will be distributed.

STC-01 | INCORRECT INPUT USED

Category	Severity	Location	Status
Volatile Code	Medium	SortedTroves.sol: 96~97	Resolved

Description

The `troveManagerHelpers` is an interface variable for the interface `ITroveManagerHelpers`. The interface is using `_troveManagerAddress` instead of `_troveManagerHelpersAddress`.

```
96        troveManagerHelpers = ITroveManagerHelpers(_troveManagerAddress);
```

Any function that calls or checks using `troveManagerHelpers` will most likely result in a failure.

Recommendation

Change `_troveManagerAddress` to `_troveManagerHelpersAddress`.

Alleviation

[DeFi Franc] :

The team heeded the advice and resolved this issue by changing the input, in the commit ca4d9ec85e38429fa484d6bb40dc7ffc9770256a.

STC-02 | LACK OF INPUT VALIDATION

Category	Severity	Location	Status
Inconsistency	Minor	SortedTroves.sol: 83~84	Resolved

Description

The input parameter `_troveManagerHelpersAddress` is missing input validation. Despite `_troveManagerAddress` and `_borrowerOperationsAddress` being checked if they are contracts via the function `checkContract`, this check is not performed on `_troveManagerHelpersAddress`. Lack of input validation can result in assigning an incorrect address or zero address.

Recommendation

Consider adding `checkContract(_troveManagerHelpersAddress)` to check if `_troveManagerHelpersAddress` is not the zero address and also a contract.

Alleviation

[DeFi Franc]

The team heeded the advice and resolved the finding in the commit hash 616f2a84a7a7a7bfdd2f042b84ffbd147dd1067b.

STD-01 INCOMPATIBLE WITH TOKENS WITH MORE THAN 18 DECIMALS

Category	Severity	Location	Status
Control Flow	Minor	Dependencies/SafetyTransfer.sol: 8~9	Resolved

Description

Although uncommon, there are tokens that have more than 18 decimals such as the `YAMv2` token, which has 24 decimals. The current implementation does not have any control flow that takes decimals greater than 18 into consideration.

Recommendation

Carefully check the decimal of assets added to the protocol in the future, or include the following code to handle assets that have more than 18 decimals:

```
19          } else {
20              return _amount.mul(10**(decimals - 18))
21          }
```

Alleviation

[DeFi Franc] :

The team heeded the advice and resolved the finding in the commit 65fd7d4fb22638ff87db63cfaccb45fb18e293e6.

TMC-01 | UNCALLABLE FUNCTION IN `TroveManager`

Category	Severity	Location	Status
Volatile Code	Medium	TroveManager.sol: 980~981	Resolved

Description

Line 980 makes an external call to the function `troveManagerHelpers.updateStakeAndTotalStakes()`. However, it is uncallable since the modifier for `updateStakeAndTotalStakes()` is `onlyBorrowerOperations` which limits the caller to the `BorrowerOperations` contract and does not include the `TroveManager` contract.

Recommendation

Change the modifier `onlyBorrowerOperations` to `onlyBOorTM`. If the suggested change is made, also consider removing the function `updateStakeAndTotalStakesTrove` in `TroveManagerHelpers` contract since the functionality will overlap with `updateStakeAndTotalStakes`.

Alleviation

[DeFi Franc] :

The team heeded the advice and resolved this issue by changing the modifier in the commit a38c2c749fe3925a0e1c218c8bed22a3ebb3f041.

TMH-01 | INCORRECT MODIFIER

Category	Severity	Location	Status
Inconsistency	Medium	TroveManagerHelpers.sol: 896~897	Resolved

Description

The comment on line 888 states : `Trove property setters, called by TroveManager` . This comment suggests that the function that follows will be called by the contract `TroveManager` .

```
888    // --- Trove property setters, called by TroveManager ---
889
890    // todo: only Trovemanager
891    function setTroveDeptAndColl(
892        address _asset,
893        address _borrower,
894        uint256 _debt,
895        uint256 _coll
896    ) external override onlyBorrowerOperations {
```

However, the function that follows is `setTroveDeptAndColl()` and has the modifier `onlyBorrowerOperations` . Furthermore, the contract `BorrowerOperations` does not contain any calls to the function `setTroveDeptAndColl()` . Instead, the contract `TroveManager` calls the function `setTroveDeptAndColl()` .

If `borrowerOperationsAddress` is properly set to the contract `BorrowerOperations` , this function will not be callable and subsequently, the redemption of collateral will not be possible since the function `redeemCollateral()` in the contract `TroveManager` calls this function.

Recommendation

Change the modifier from `_onlyBorrowerOperations` to `onlyTroveManager` .

Alleviation

[DeFi Franc] :

The team heeded the advice and resolved this issue by changing the modifier, in the commit 4903708fcf4e59eaa82c191728ca2d1d8bacdb44.

TMH-02 | UNCALLABLE FUNCTIONS IN `TroveManagerHelpers`

Category	Severity	Location	Status
Volatile Code	Medium	TroveManagerHelpers.sol: 471~472, 473~474, 971~972	Resolved

Description

The following lines of code in the contract `TroveManagerHelpers` are uncallable due to the modifier of the external callee function.

- Line 471 is assumed to make an external call to the function `decreaseDCHFDebt()` of the `ActivePool` contract. However the function `decreaseDCHFDebt()` has a modifier `callerIsBOorTroveMorSP` which requires the caller to be `BorrowerOperations`, `TroveManager`, or `StabilityPool` contract.
- Line 473 is assumed to make an external call to the function `sendAsset` of the `ActivePool` contract. However the function `sendAsset` has a modifier `callerIsBOorTroveMorSP` which requires the caller to be `BorrowerOperations`, `TroveManager`, or `StabilityPool` contract.
- Line 971 is assumed to make an external call to the function `increaseDCHFDebt` of the `ActivePool` contract. However the function `increaseDCHFDebt` has a modifier `callerIsBOorTroveM` which requires the caller to be `BorrowerOperations` or `TroveManager` contract.

Since `TroveManagerHelpers` is not included in any of the modifiers of the callee function, it prevents the `TroveManagerHelpers` contract from calling these functions.

Recommendation

Check if the call to the external functions are correct and if it is, include the `TroveManagerHelpers` in the modifier `callerIsBOorTroveMorSP` and `callerIsBOorTroveM` of the `ActivePool` contract, to allow calls to the function from the `TroveManagerHelpers` contract.

Alleviation

[DeFi Franc] :
The team heeded the advice and resolved this issue by adding `troveManagerHelpersAddress` in the `callerIsBOorTroveMorSP` modifier, in the commit cc00b0ecb22f7b435bbef88f14ed4645db4c972d.

TMH-03 | LACK OF INPUT VALIDATION

Category	Severity	Location	Status
Inconsistency	Minor	TroveManagerHelpers.sol: 119~120	Resolved

Description

The input `_troveManagerAddress` does not have any input validation. Other input parameters are passed into the function `checkContract()` which checks if the address is not the zero address and if the address contains code.
Multiple functions have the `onlyTroveManager` modifier which requires the caller to be a `troveManager` contract and these functions will not be callable if the address is incorrectly set.

Recommendation

Consider adding `checkContract(_troveManagerAddress)` to prevent incorrectly setting `troveManager` as a zero address or to an EOA.

Alleviation

[DeFi Franc] :
The team heeded the advice and resolved the finding in the commit ca4d9ec85e38429fa484d6bb40dc7ffc9770256a.

CKP-04 | REDUNDANT CODE COMPONENTS

Category	Severity	Location	Status
Volatile Code	Informational	BorrowerOperations.sol: 626, 747; StabilityPool.sol: 812	Resolved

Description

The linked statements do not affect the functionality of the codebase and appear to be either leftovers from test code or older functionality.

Recommendation

We advise to remove the redundant statements for production environments.

Alleviation

[DeFi Franc]

The team heeded the advice and resolved the finding in the commit f74f1f5017dffcdfb502efb93c539f379406d8e7.

CKP-05 | MISSING ERROR MESSAGES

Category	Severity	Location	Status
Coding Style	Informational	AdminContract.sol: 41; DfrancParameters.sol: 77; MON/CommunityIssuance.sol: 84; MON/MONStaking.sol: 102; PriceFeed.sol: 40, 51	Resolved

Description

The **require** can be used to check for conditions and throw an exception if the condition is not met. It is better to provide a string message containing details about the error that will be passed back to the caller.

Recommendation

We advise adding error messages to the linked **require** statements.

Alleviation

[DeFi Franc] :
The team heeded the advice and resolved the finding in the commit hash 3946dc9f1ac0b57b6da9ccb2ab552e1aca167069.

CKP-06 | MISSING EMIT EVENTS

Category	Severity	Location	Status
Coding Style	Informational	AdminContract.sol: 31, 64; BorrowerOperations.sol: 747; CollSurplusPool.sol: 122, 129, 133; DCHFToken.sol: 179; DfrancParameters.sol: 76, 94, 98, 126; MON/CommunityIssuance.sol: 83, 88, 96, 116, 142, 228; MON/LockedMON.sol: 40, 48, 80, 98, 114, 140; MON/MONStaking.sol: 188, 192, 332; Migrations.sol: 17, 21; PriceFeed.sol: 50; SortedTroves.sol: 526, 533; StabilityPool.sol: 784, 791; StabilityPoolManager.sol: 32, 41, 58; TroveManager.sol: 48, 48, 1043, 1047, 1051; TroveManagerHelpers.sol: 80, 89, 98, 114, 114	Partially Resolved

Description

There should always be events emitted in the sensitive functions that are controlled by centralization roles.

Recommendation

It is recommended emitting events for the sensitive functions that are controlled by centralization roles.

Alleviation

[DeFi Franc]

The team heeded the advice and partially resolved the finding in the commit hash f74e188e42b3e71e51f25f10fafef276a92c6fd4.

CKP-07 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	Informational	MON/MONStaking.sol: 197; Proxy/ETHTransferScript.sol: 7	Resolved

Description

Addresses should be checked before assignment or external call to make sure they are not zero addresses.

```
197         treasury = _treasury;
```

- `_treasury` is not zero-checked before being used.

```
7         (bool success, ) = _recipient.call{ value: _amount }("");
```

- `_recipient` is not zero-checked before being used.

Recommendation

Add a zero-check for the passed-in address value to prevent unexpected errors.

Alleviation

[DeFi Franc] :

The team heeded the advice and resolved this issue by adding the Zero address verification in the `MONStaking` contract, and by deleting the `ETHTransferScript` contract, in the commit f74e188e42b3e71e51f25f10fafef276a92c6fd4.

TMH-04 | REPETITIVE FUNCTION IMPLEMENTATION

Category	Severity	Location	Status
Coding Style	Informational	TroveManagerHelpers.sol: 349~350, 373~374	Resolved

Description

The following functions in `TroveManagerHelpers` have different function names but have the same logic implemented.

- Both the functions `removeStake()` and `removeStakeTrove()` call the internal function `_removeStake()` with the only difference being the modifier.
- Both the functions `updateStakeAndTotalStakes()` and `updateStakeAndTotalStakesTrove()` call the internal function `_updateStakeAndTotalStakes()` with the only difference being the modifier.

The modifier for the functions `removeStake` and `updateStakeAndTotalStakes` is `onlyBOorTM`, while the modifier for the functions `removeStakeTrove` and `updateStakeAndTotalStakesTrove` is `onlyTroveManager`.

Since the `TroveManager` contract can call the function with either modifier and the effect has no difference, `removeStakeTrove` and `updateStakeAndTotalStakesTrove` can be removed.

Recommendation

Consider removing the functions `removeStakeTrove()` and `updateStakeAndTotalStakesTrove()`. Furthermore, change any calls to the function `removeStakeTrove()` to the function `removeStake()` and calls to the function `updateStakeAndTotalStakesTrove()` to `updateStakeAndTotalStakes()`. This will improve the maintainability and readability of the code.

Alleviation

[DeFi Franc] :
The team heeded the advice and resolved this issue in the commit 80c9ec708ee778e24e71d49daaf937692178727e.

OPTIMIZATIONS | DEFI FRANC

ID	Title	Category	Severity	Status
GLOBAL-03	Unnecessary Use Of SafeMath And SafeMathUpgradeable	Gas Optimization	Optimization	Acknowledged
BOC-01	Useless Statement	Logical Issue	Optimization	Resolved
CKP-03	Improper Usage Of `public` And `external` Type	Gas Optimization	Optimization	Resolved
DMD-01	Unnecessary Write To Memory	Gas Optimization	Optimization	Resolved
LMO-02	Costly Operation Inside Loop	Gas Optimization	Optimization	Resolved
MOT-02	State Variable Should Be Declared Constant	Gas Optimization	Optimization	Resolved
TMC-02	Unnecessary External Call	Gas Optimization	Optimization	Resolved

GLOBAL-03 UNNECESSARY USE OF SAFEMATH AND SAFEMATHUPGRADEABLE

Category	Severity	Location	Status
Gas Optimization	Optimization		Acknowledged

Description

The `SafeMath` and `SafeMathUpgradeable` library is used unnecessarily throughout the codebase. With Solidity compiler versions 0.8.0 or newer, arithmetic operations will automatically revert in case of integer overflow or underflow.

Recommendation

We advise removing the usage of `SafeMath` and `SafeMathUpgradeable` library and using the built-in arithmetic operations provided by the Solidity programming language for gas optimization and code clarity.

Alleviation

[DeFi Franc]

The team acknowledged the finding and decided to keep the codebase unchanged.

BOC-01 | USELESS STATEMENT

Category	Severity	Location	Status
Logical Issue	Optimization	BorrowerOperations.sol: 191	Resolved

Description

In the `openTrove()` function of the `BorrowerOperations` contract, a line is present while performing no action.

```
191         vars.DCHFFee;
```

Recommendation

It is recommended to remove this line if it is not necessary.

Alleviation

[DeFi Franc]

The team heeded the advice and resolved the finding in the commit hash fe05bb58171d5e4195c82ec4ed82ca6beb5392e4.

CKP-03 | IMPROPER USAGE OF `public` AND `external` TYPE

Category	Severity	Location	Status
Gas Optimization	Optimization	DCHFToken.sol: 106, 111; Dependencies/ERC20Permit.sol: 82; DfrancParameters.sol: 126; MON/LockedMON.sol: 40, 80, 98, 114; MON/MONStaking.sol: 196; Migrations.sol: 17, 21; TroveManagerHelpers.sol: 782	Resolved

Description

`public` functions that are never called by the contract could be declared as `external`. `external` functions are more efficient than `public` functions.

Recommendation

Consider using the external attribute for public functions that are never called within the contract.

Alleviation

[DeFi Franc]

The team heeded the advice and resolved the finding in the commit hash f8f67806778ac36578d9fea42abc0fbb881694a8.

DMD-01 | UNNECESSARY WRITE TO MEMORY

Category	Severity	Location	Status
Gas Optimization	Optimization	Dependencies/DfrancMath.sol: 113~114	Resolved

Description

The collateral ratio is first stored in a memory variable `newCollRatio`. Since the variable is not used and the function name conveys what the value returned is, the calculation can be returned directly. This will save the gas consumed from writing and reading to memory.

Recommendation

Consider removing line 111 and changing line 113 to:

```
113             return _coll.mul(_price).div(_debt);
```

Alleviation

[DeFi Franc]

The team heeded the advice and resolved the finding in the commit hash 8865e45b98fd6f54e3890e60b1361d73668b1dba.

LMO-02 | COSTLY OPERATION INSIDE LOOP

Category	Severity	Location	Status
Gas Optimization	Optimization	MON/LockedMON.sol: 64	Resolved

Description

Accessing storage variables in a loop can be costly in terms of gas consumption.

```
64              assignedMONTokens += _totalSupply;
```

Recommendation

We recommend using a local variable to hold the intermediate result.

Alleviation

[DeFi Franc]

The team heeded the advice and resolved the finding in the commit hash 3ec2be6920b68412bf7b22c8174f58462a959483.

MOT-02 | STATE VARIABLE SHOULD BE DECLARED CONSTANT

Category	Severity	Location	Status
Gas Optimization	Optimization	MON/MONToken.sol: 12	Resolved

Description

State variables that never change should be declared as `constant` to save gas.

```
12      uint256 internal _1_MILLION = 1e24; // 1e6 * 1e18 = 1e24
```

- `_1_MILLION` should be declared `constant` .

Recommendation

We recommend adding the `constant` attribute to state variables that never change.

Alleviation

[DeFi Franc] :
The team heeded the advice and resolved this issue by putting the variable as constant, in the commit c5463c14597b0bccf22dc49ced8fb00c2c9aafd6.

TMC-02 | UNNECESSARY EXTERNAL CALL

Category	Severity	Location	Status
Gas Optimization	Optimization	TroveManager.sol: 648~649	Resolved

Description

The function `troveManagerHelpers.checkRecoveryMode()` will make an external call to the contract `troveManagerHelpers` and ultimately invoke the internal function `_checkRecoveryMode`. However, `_checkRecoveryMode` is implemented in `DfrancBase.sol`, which is also a base contract for the contract `troveManager`. Since `_checkRecoveryMode` can be called internally, the external call to `troveManagerHelpers` is unnecessary.

Recommendation

Consider changing the code on line 648 to the following:

```
648         vars.recoveryModeAtStart = _checkRecoveryMode(_asset, vars.price);
```

Using internal calls will reduce gas costs compared to making external calls.

Alleviation

[DeFi Franc]

The team heeded the advice and resolved the finding in the commit hash f4e31a2e04150257f1a87a78d106a0ce10b957e4.

APPENDIX | DEFI FRANC

Finding Categories

Categories	Description
Centralization / Privilege	Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.
Gas Optimization	Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.
Mathematical Operations	Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.
Logical Issue	Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.
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.
Language Specific	Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.
Coding Style	Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.
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.

Checksum Calculation Method

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

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

DISCLAIMER | CERTIK

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

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

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

FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

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

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

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

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

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

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

CertiK | Securing the Web3 World

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