Security Assessment

Flux

Apr 22nd, 2021

Summary

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

The auditing process pays special attention to the following considerations:

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

Additionally, this audit is based on a premise that all external smart contracts are implemented safely. For example, we assumed that the chainlink oracle will be available and reliable.

The security assessment resulted in 15 findings that ranged from minor to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	Flux
Description	Lending Platform
Platform	BSC
Language	Solidity
Codebase	https://github.com/01-finance/flux-protocol/tree/bsc/testnet
Commits	d8287ee4534fa4d5cf8c48320229e3d837272614

Audit Summary

Delivery Date	Apr 22, 2021
Audit Methodology	Manual Review
Key Components	

Vulnerability Summary

Total Issues	15
Critical	0
Major	0
Minor	7
Informational	8
Discussion	0

Audit Scope

ID	file	SHA256 Checksum
FAI	FluxApp.sol	5fe7132918a07b1a59531170871a9bfada4e581dff13016a1d084036200436f1
FMI	FluxMint.sol	36899b5eaed77a0821b3c3c79dae6a0152d9736946ad02ed497018a90a4395aa
GIV	Guard.sol	16a4b3fdf5288eef6677ff545fdf7ae2ee4430be2ed798d82b1e2ef4f1c31272
LOI	LinkOracle.sol	15e442c6175a1d5eec9458edf3bff04f043c87f5df5d9ffb0fa570280fee84cc
MIV	Market.sol	ff8c39895769da4a844777a3c3fa8f79de135c83458dc9ecb6d5c6395d100bcd
MCF	MarketCFX.sol	08e62ccbd99cb525072442fcdf50a52e8195ec1f3a66a1d291be526fe6953403
MER	MarketERC20.sol	1cec73911eb66143129eaa0e400a74f9959acd4a012e773aa7289abba301fb8b

Centralization Roles

The Flux smart contract introduced an authorization.

Owner

[FluxApp.sol]

- setConfig()
- approveMarket()
- resetCollRatio()
- removeMarket()
- changeSupplyStatus()
- changeBorrowStatus()
- changeLiquidateStatus()
- changeAllActionStatus()
- setMarketStatus()
- setBorrowLimit()
- setFluxMint()
- changeWeights()
- stakePoolApprove()
- setStakePoolStatus()
- removeStakePool()

[FluxMint.sol]

- resetTeamAdress()
- grantFlux()
- changeWeights()
- setPoolSeed()
- removePool()
- refreshFluxMintIndex()
- settleOnce()

[Guard.sol]

- withdraw()

[LinkOracle.sol]

- setAggregator()

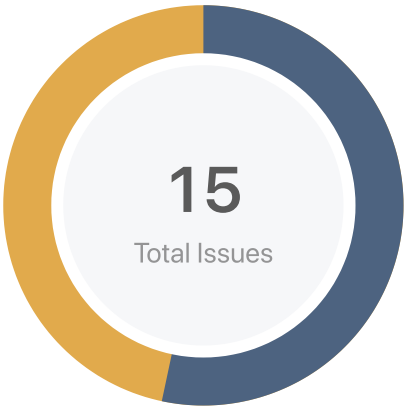
[Market.sol]

- changeOracle()

[MarketCFX.sol]

- initWithdrawProxy()

Findings



Critical	0 (0.00%)
Major	0 (0.00%)
Minor	7 (46.67%)
Informational	8 (53.33%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
FAI-01	Missing Some Important Checks	Logical Issue	Informational	Partially Resolved
FAI-02	**Overly-Privilege Granted To Governance**	**Centralization / Privilege**	**Minor**	**Acknowledged**
FAI-03	Missing Emit Events	Coding Style	Informational	Resolved
FMI-01	Missing Some Important Checks	Logical Issue	Informational	Partially Resolved
FMI-02	Missing Return Value	Logical Issue	Informational	Resolved
FMI-03	**Overly-Privilege Granted To Governance**	**Centralization / Privilege**	**Minor**	**Acknowledged**
FMI-04	Missing Emit Events	Coding Style	Informational	Resolved
GIV-01	Missing Some Important Checks	Logical Issue	Informational	Partially Resolved
GIV-02	**Overly-Privilege Granted To Governance**	**Centralization / Privilege**	**Minor**	**Acknowledged**
LOI-01	**Overly-Privilege Granted To Governance**	**Centralization / Privilege**	**Minor**	**Acknowledged**
MCF-01	**Overly-Privilege Granted To Governance**	**Centralization / Privilege**	**Minor**	**Acknowledged**
MCF-02	Missing Emit Events	Coding Style	Informational	Resolved

ID	Title	Category	Severity	Status
MIV-01	Duplicated Transfer	Logical Issue	Minor	Resolved
MIV-02	**Overly-Privilege Granted To Governance**	**Centralization / Privilege**	**Minor**	**Acknowledged**
MIV-03	Missing Emit Events	Coding Style	Informational	Resolved

FAI-01 | Missing Some Important Checks

Category	Severity	Location	Status
Logical Issue	Informational	FluxApp.sol: 70	Partially Resolved

Description

Function `initialize()` in contract `FluxMint.sol` is missing parameter address zero check.

Function `initialize()` in contract `Guard.sol` is missing parameter address zero check.

Function `initialize()` in contract `FluxApp.sol` is missing parameter address zero check.

Function `changeWeights()` in contract `FluxMint.sol` is missing upper limit check for different `weight`.

Function `mint()` in contract `MarketCFX.sol` is missing parameter value zero check.

Zero check is applicable to other similar places.

Recommendation

Consider adding necessary check. For example:

```
function initialize(address admin_, address fluxAPP_) external initializer {
    require(admin_ != address(0), "FluxMint: admin_ is zero address");
    require(fluxAPP_ != address(0), "FluxMint: fluxAPP_ is zero address");
    ......
}
function changeWeights(
        uint16 borrow,
        uint16 supply,
        uint16 team,
        uint16 community
    ) external onlyAppOrAdmin {
      require(borrow+supply+team+community <= WEIGHT_UNIT);
        ......
}
function mint() external payable override {
        require(msg.value > 0, "REPAY_IS_ZERO");
        _supply(msg.sender, msg.value);
}
```

Alleviation

The team heeded some of our advice and changed related codes. Code change was applied in commit 1d14c73fcc24853cc76f7202e21c89b5ec2f575e.

FAI-02 | Overly-Privilege Granted To Governance

Category	Severity	Location	Status
Centralization / Privilege	Minor	FluxApp.sol: 79, 92, 118, 136, 344, 354, 363, 373, 385, 398, 629, 643, 712, 721, 736	Acknowledged

Description

In Flux, `owner` is a role with high privilege. `owner` has ability to set many sensitive protocol parameters like `configs`, `collRatioMan`, `liquidateDisabled`, `teamFluxReceive`, `oracle`, `withdrawProxy`, `aggregators` etc. Besides, `owner` also has the ability to transfer some assets. We believe that by using such permission setups, Flux could get a short-term benefit for rapid development and stable operation. However, if this role was granted to a malicious person accidentally, system/users assets may suffer huge losses. For example, the `owner` can withdraw assets from `Guard`. The `owner` can gain benefits by influencing the `underlying` price. Flux project will be eventually hurt without community engagement and adoption.

Recommendation

Consider adding time lock to all sensitive functions and using multi-signature where critical.

Alleviation

Flux responded that they will add a timelock.

FAI-03 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	FluxApp.sol: 398, 629, 721, 736	Resolved

Description

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

Recommendation

Consider adding an emit to these functions.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit 1d14c73fcc24853cc76f7202e21c89b5ec2f575e.

FMI-01 | Missing Some Important Checks

Category	Severity	Location	Status
Logical Issue	Informational	FluxMint.sol: 109, 156	Partially Resolved

Description

Function `initialize()` in contract `FluxMint.sol` is missing parameter address zero check.

Function `initialize()` in contract `Guard.sol` is missing parameter address zero check.

Function `initialize()` in contract `FluxApp.sol` is missing parameter address zero check.

Function `changeWeights()` in contract `FluxMint.sol` is missing upper limit check for different `weight`.

Function `mint()` in contract `MarketCFX.sol` is missing parameter value zero check.

Zero check is applicable to other similar places.

Recommendation

Consider adding necessary check. For example:

```
function initialize(address admin_, address fluxAPP_) external initializer {
    require(admin_ != address(0), "FluxMint: admin_ is zero address");
    require(fluxAPP_ != address(0), "FluxMint: fluxAPP_ is zero address");
    ......
}
function changeWeights(
        uint16 borrow,
        uint16 supply,
        uint16 team,
        uint16 community
    ) external onlyAppOrAdmin {
      require(borrow+supply+team+community <= WEIGHT_UNIT);
        ......
}
function mint() external payable override {
        require(msg.value > 0, "REPAY_IS_ZERO");
        _supply(msg.sender, msg.value);
}
```

Alleviation

The team heeded some of our advice and changed related codes. Code change was applied in commit 1d14c73fcc24853cc76f7202e21c89b5ec2f575e.

FMI-02 | Missing Return Value

Category	Severity	Location	Status
Logical Issue	Informational	FluxMint.sol: 310	Resolved

Description

Function `_unlockDAOFlux()` declared a return value of `bool`, but within its function body, there is no value returned.

Recommendation

Consider removing return declaration or returning a bool value within the function body.

Alleviation

The team heeded our advice and changed related code. Code change was applied in commit 1d14c73fcc24853cc76f7202e21c89b5ec2f575e.

FMI-03 | Overly-Privilege Granted To Governance

Category	Severity	Location	Status
Centralization / Privilege	Minor	FluxMint.sol: 131, 142, 171, 181, 232, 155, 164	Acknowledged

Description

In Flux, `owner` is a role with high privilege. `owner` has ability to set many sensitive protocol parameters like `configs`, `collRatioMan`, `liquidateDisabled`, `teamFluxReceive`, `oracle`, `withdrawProxy`, `aggregators` etc. Besides, `owner` also has the ability to transfer some assets. We believe that by using such permission setups, Flux could get a short-term benefit for rapid development and stable operation. However, if this role was granted to a malicious person accidentally, system/users assets may suffer huge losses. For example, the `owner` can withdraw assets from `Guard`. The `owner` can gain benefits by influencing the `underlying` price. Flux project will be eventually hurt without community engagement and adoption.

Recommendation

Consider adding time lock to all sensitive functions and using multi-signature where critical.

Alleviation

Flux responded that they will add a timelock.

FMI-04 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	FluxMint.sol: 131	Resolved

Description

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

Recommendation

Consider adding an emit to these functions.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit 1d14c73fcc24853cc76f7202e21c89b5ec2f575e.

GIV-01 | Missing Some Important Checks

Category	Severity	Location	Status
Logical Issue	Informational	Guard.sol: 60	Partially Resolved

Description

Function `initialize()` in contract `FluxMint.sol` is missing parameter address zero check.

Function `initialize()` in contract `Guard.sol` is missing parameter address zero check.

Function `initialize()` in contract `FluxApp.sol` is missing parameter address zero check.

Function `changeWeights()` in contract `FluxMint.sol` is missing upper limit check for different `weight`.

Function `mint()` in contract `MarketCFX.sol` is missing parameter value zero check.

Zero check is applicable to other similar places.

Recommendation

Consider adding necessary check. For example:

```
function initialize(address admin_, address fluxAPP_) external initializer {
    require(admin_ != address(0), "FluxMint: admin_ is zero address");
    require(fluxAPP_ != address(0), "FluxMint: fluxAPP_ is zero address");
    ......
}
function changeWeights(
        uint16 borrow,
        uint16 supply,
        uint16 team,
        uint16 community
    ) external onlyAppOrAdmin {
      require(borrow+supply+team+community <= WEIGHT_UNIT);
        ......
}
function mint() external payable override {
        require(msg.value > 0, "REPAY_IS_ZERO");
        _supply(msg.sender, msg.value);
}
```

Alleviation

The team heeded some of our advice and changed related codes. Code change was applied in commit 1d14c73fcc24853cc76f7202e21c89b5ec2f575e.

GIV-02 | Overly-Privilege Granted To Governance

Category	Severity	Location	Status
Centralization / Privilege	Minor	Guard.sol: 106	Acknowledged

Description

In Flux, `owner` is a role with high privilege. `owner` has ability to set many sensitive protocol parameters like `configs`, `collRatioMan`, `liquidateDisabled`, `teamFluxReceive`, `oracle`, `withdrawProxy`, `aggregators` etc. Besides, `owner` also has the ability to transfer some assets. We believe that by using such permission setups, Flux could get a short-term benefit for rapid development and stable operation. However, if this role was granted to a malicious person accidentally, system/users assets may suffer huge losses. For example, the `owner` can withdraw assets from `Guard`. The `owner` can gain benefits by influencing the `underlying` price. Flux project will be eventually hurt without community engagement and adoption.

Recommendation

Consider adding time lock to all sensitive functions and using multi-signature where critical.

Alleviation

Flux responded that they will add a timelock.

LOI-01 | Overly-Privilege Granted To Governance

Category	Severity	Location	Status
Centralization / Privilege	Minor	LinkOracle.sol: 11	Acknowledged

Description

In Flux, `owner` is a role with high privilege. `owner` has ability to set many sensitive protocol parameters like `configs`, `collRatioMan`, `liquidateDisabled`, `teamFluxReceive`, `oracle`, `withdrawProxy`, `aggregators` etc. Besides, `owner` also has the ability to transfer some assets. We believe that by using such permission setups, Flux could get a short-term benefit for rapid development and stable operation. However, if this role was granted to a malicious person accidentally, system/users assets may suffer huge losses. For example, the `owner` can withdraw assets from `Guard`. The `owner` can gain benefits by influencing the `underlying` price. Flux project will be eventually hurt without community engagement and adoption.

Recommendation

Consider adding time lock to all sensitive functions and using multi-signature where critical.

Alleviation

Flux responded that they will add a timelock.

MCF-01 | Overly-Privilege Granted To Governance

Category	Severity	Location	Status
Centralization / Privilege	Minor	MarketCFX.sol: 32	Acknowledged

Description

In Flux, `owner` is a role with high privilege. `owner` has ability to set many sensitive protocol parameters like `configs`, `collRatioMan`, `liquidateDisabled`, `teamFluxReceive`, `oracle`, `withdrawProxy`, `aggregators` etc. Besides, `owner` also has the ability to transfer some assets. We believe that by using such permission setups, Flux could get a short-term benefit for rapid development and stable operation. However, if this role was granted to a malicious person accidentally, system/users assets may suffer huge losses. For example, the `owner` can withdraw assets from `Guard`. The `owner` can gain benefits by influencing the `underlying` price. Flux project will be eventually hurt without community engagement and adoption.

Recommendation

Consider adding time lock to all sensitive functions and using multi-signature where critical.

Alleviation

Flux responded that they will add a timelock.

MCF-02 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	MarketCFX.sol: 32	Resolved

Description

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

Recommendation

Consider adding an emit to these functions.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit 1d14c73fcc24853cc76f7202e21c89b5ec2f575e.

MIV-01 | Duplicated Transfer

Category	Severity	Location	Status
Logical Issue	Minor	Market.sol: 406	Resolved

Description

According to the logic of function `liquidate()`. When users liquidate borrower's debt, they need to transfer `underlying` to `Guard`. Related market `mkt` apply unlimited allowance of `Guard`. Then function `liquidate()` of `Market` is called. It seems `Market` will get these related `underlying` from `Guard`. However, `msg.sender` which represent users is passed as the `liquidator`, according to the logic of function `liquidate()` of `Market`, `Market` will get money from `msg.sender` again.

Recommendation

Consider correcting the logic of function `liquidate()` of `Market`. Pass `msg.sender` to `underlyingTransferIn()` but not `liquidator`.

Alleviation

The team heeded our advice and changed related code. Code change was applied in commit 1d14c73fcc24853cc76f7202e21c89b5ec2f575e.

MIV-02 | Overly-Privilege Granted To Governance

Category	Severity	Location	Status
Centralization / Privilege	Minor	Market.sol: 145	Acknowledged

Description

In Flux, `owner` is a role with high privilege. `owner` has ability to set many sensitive protocol parameters like `configs`, `collRatioMan`, `liquidateDisabled`, `teamFluxReceive`, `oracle`, `withdrawProxy`, `aggregators` etc. Besides, `owner` also has the ability to transfer some assets. We believe that by using such permission setups, Flux could get a short-term benefit for rapid development and stable operation. However, if this role was granted to a malicious person accidentally, system/users assets may suffer huge losses. For example, the `owner` can withdraw assets from `Guard`. The `owner` can gain benefits by influencing the `underlying` price. Flux project will be eventually hurt without community engagement and adoption.

Recommendation

Consider adding time lock to all sensitive functions and using multi-signature where critical.

Alleviation

Flux responded that they will add a timelock.

MIV-03 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	Market.sol: 145	Resolved

Description

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

Recommendation

Consider adding an emit to these functions.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit 1d14c73fcc24853cc76f7202e21c89b5ec2f575e.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Mathematical Operations

Mathematical Operation exhibits entail findings that relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings are exhibits that detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

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

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a struct assignment operation affecting an in-memory struct rather than an in storage one.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete .

Coding Style

Coding Style findings usually do not affect the generated byte-code and comment on how to make the codebase more legible and as a result easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as constant contract variables aiding in their legibility and maintainability.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

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

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

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

About

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