DFORCE LENDING V2 SECURITY AUDIT REPORT

Nov 07, 2024

MixBytes()

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only. The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of the Client. If you are not the intended recipient(s) of this document, please note that any disclosure, copying or dissemination of its content is strictly forbidden.

1.2 Security Assessment Methodology

A group of auditors are involved in the work on the audit. The security engineers check the provided source code independently of each other in accordance with the methodology described below:

1. Project architecture review:

- Project documentation review.
- General code review.
- Reverse research and study of the project architecture on the source code alone.

Stage goals
- Build an independent view of the project's architecture.
- Identifying logical flaws.

2. Checking the code in accordance with the vulnerabilities checklist:

- Manual code check for vulnerabilities listed on the Contractor's internal checklist. The Contractor's checklist is constantly updated based on the analysis of hacks, research, and audit of the clients' codes.
- Code check with the use of static analyzers (i.e Slither, Mythril, etc).

Stage goal

Eliminate typical vulnerabilities (e.g. reentrancy, gas limit, flash loan attacks etc.).

3. Checking the code for compliance with the desired security model:

- Detailed study of the project documentation.
- Examination of contracts tests.
- Examination of comments in code.
- Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit.
- Exploits PoC development with the use of such programs as Brownie and Hardhat.

Stage goal

Detect inconsistencies with the desired model.

4. Consolidation of the auditors' interim reports into one:

- Cross check: each auditor reviews the reports of the others.
- Discussion of the issues found by the auditors.
- Issuance of an interim audit report.

Stage goals

- Double-check all the found issues to make sure they are relevant and the determined threat level is correct.
- Provide the Client with an interim report.

5. Bug fixing & re-audit:

- The Client either fixes the issues or provides comments on the issues found by the auditors. Feedback from the Customer must be received on every issue/bug so that the Contractor can assign them a status (either "fixed" or "acknowledged").
- Upon completion of the bug fixing, the auditors double-check each fix and assign it a specific status, providing a proof link to the fix.
- A re-audited report is issued.

Stage goals

- Verify the fixed code version with all the recommendations and its statuses.
- Provide the Client with a re-audited report.

6. Final code verification and issuance of a public audit report:

- The Customer deploys the re-audited source code on the mainnet.
- The Contractor verifies the deployed code with the re-audited version and checks them for compliance.
- If the versions of the code match, the Contractor issues a public audit report.

Stage goals

- Conduct the final check of the code deployed on the mainnet.
- Provide the Customer with a public audit report.

Finding Severity breakdown

All vulnerabilities discovered during the audit are classified based on their potential severity and have the following classification:

Severity	Description
Critical	Bugs leading to assets theft, fund access locking, or any other loss of funds.
High	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.
Medium	Bugs that can break the intended contract logic or expose it to DoS attacks, but do not cause direct loss funds.
Low	Bugs that do not have a significant immediate impact and could be easily fixed.

Based on the feedback received from the Customer regarding the list of findings discovered by the Contractor, they are assigned the following statuses:

Status	Description
Fixed	Recommended fixes have been made to the project code and no longer affect its security.
Acknowledged	The Customer is aware of the finding. Recommendations for the finding are planned to be resolved in the future.

1.3 Project Overview

The dForce project is a decentralized lending protocol. Users can supply collateral, borrow assets and earn interest. The features of the dForce protocol are:

- SuperCharged mode - categorizing some assets by having corellated prices to allow less collaterization ratios;
- liquidation threshold - a buffer between the maximum borrowing power and insolvency
- Segregated mode - limit risks of collaterization by some assets;
- delay payment - timelock on transfer-outs if some conditions are met.

1.4 Project Dashboard

Project Summary

Title	Description
Client	dForce
Project name	Lending V2
Timeline	06.09.2023 - 17.06.2024
Number of Auditors	3

Project Log

Date	Commit Hash	Note
06.09.2023	6f3a7b6946d8226b38e7f0d67a50e68a28fe5165	Initial commit for the audit
14.11.2023	abf7ef8d327a15a9e5e5f8bec6b444142d988f34	Commit for the re-audit
29.11.2023	490a30f5a2e0e369f9ea52097b28254f11c5ada6	Commit for the re-audit 2
13.06.2024	5d005d16a96499828a6703f41cda2b946887800e	Commit for the diff audit

Project Scope

The audit covered the following files:

File name	Link
Controller.sol	Controller.sol
ControllerStorage.sol	ControllerStorage.sol

File name	Link
ControllerV2ExtraBase.sol	ControllerV2ExtraBase.sol
ControllerV2ExtraExplicit.sol	ControllerV2ExtraExplicit.sol
ControllerV2ExtraImplicit.sol	(ControllerV2ExtraImplicit.sol
ControllerV2.sol	ControllerV2.sol
DefaultTimeLock.sol	DefaultTimeLock.sol
iETH.sol	iETH.sol
iETHV2.sol	iETHV2.sol
iToken.sol	iToken.sol
iTokenV2.sol	iTokenV2.sol
RewardDistributorSecondV3.sol	RewardDistributorSecondV3.sol
RewardDistributorV3.sol	RewardDistributorV3.sol
TimeLockStrategy.sol	TimeLockStrategy.sol
TokenBase/Base.sol	Base.sol
TokenBase/InterestUnit.sol	InterestUnit.sol
TokenBase/TokenAdmin.sol	TokenAdmin.sol
TokenBase/TokenERC20.sol	TokenERC20.sol
TokenBase/TokenEvent.sol	TokenEvent.sol
TokenBase/TokenStorage.sol	TokenStorage.sol

Deployments

Base:mainnet

File name	Contract deployed	Comment
Timelock.sol	0xa4e5eb...c2d482cc	
TransparentUpgradeableProxy.sol	0xBae8d1...135fcE22	proxy for ControllerV2
ControllerV2.sol	0xc7d598...3ddc7d63	implementation
ControllerV2ExtraImplicit.sol	0x95c06b...13C86691	
ControllerV2ExtraExplicit.sol	0xd556fb...e1613EA4	
TransparentUpgradeableProxy.sol	0xD614E4...8b7D8B1d	proxy for DefaultTimeLock
DefaultTimeLock.sol	0x28bbd5...2240ef66	implementation
TransparentUpgradeableProxy.sol	0x4ca6A6...7d9E2E3F	proxy for TimeLockStrategy
TimeLockStrategy.sol	0xeae18e...b9e9a92f	implementation
TransparentUpgradeableProxy.sol	0x76B5f3...88Dc8Bba	proxy for iETHV2
iETHV2.sol	0x0d66fa...f036451f	implementation
TransparentUpgradeableProxy.sol	0xf8fBD6...2645902f	proxy for iTokenV2 iwstETH
TransparentUpgradeableProxy.sol	0x6D9Ce3...83cf4a50	proxy for iTokenV2 icbETH
TransparentUpgradeableProxy.sol	0xBb8163...22c9a2FD	proxy for iTokenV2 iUSDC
TransparentUpgradeableProxy.sol	0x82AFc9...F0E605df	proxy for iTokenV2 iUSX

File name	Contract deployed	Comment
iTokenV2.sol	0x66d7c9...0a2be001	implementation
TransparentUpgradeableProxy.sol	0xE08020...F5035d75	proxy for RewardDistributorSecondV3.sol
RewardDistributorSecondV3.sol	0x251687...8a4dff7d	implementation

Ethereum:mainnet

File name	Contract deployed	Comment
TimeLock.sol	0x17e66B...94279b94	
TransparentUpgradeableProxy.sol	0x8B53Ab...815Ad113	Proxy for ControllerV2.sol
ControllerV2.sol	0xfAEA99...1e7b1128	Implementation
ControllerV2ExtraExplicit.sol	0x69E979...44cfD832	
ControllerV2ExtraImplicit.sol	0xDDcdf0...b27fb397	
TransparentUpgradeableProxy.sol	0x5ACD75...8bfaAbc0	Proxy for iETH
iETHV2.sol	0xF42661...FD945555	Implementation
TransparentUpgradeableProxy.sol	0x8fAeF8...f1819564	Proxy for RewardDistributorSecondV3.sol
RewardDistributorSecondV3.sol	0xE6b50f...62a72bB9	Implementation
TransparentUpgradeableProxy.sol	0x3e5CB9...e0DC3D0e	Proxy for iAAVE
TransparentUpgradeableProxy.sol	0x24677e...dc889FCe	Proxy for iBUSD

File name	Contract deployed	Comment
TransparentUpgradeableProxy.sol	0xe39672...687CBC09	Proxy for iCRV
TransparentUpgradeableProxy.sol	0x298f24...68E3c3A8	Proxy for iDAI
TransparentUpgradeableProxy.sol	0xb3dc74...34d7b239	Proxy for iDF
TransparentUpgradeableProxy.sol	0x44c324...0C2dcFbF	Proxy for iEUX
TransparentUpgradeableProxy.sol	0x47C19A...9eF4022F	Proxy for iFEI
TransparentUpgradeableProxy.sol	0x71173e...Cc7E8F63	Proxy for iFRAX
TransparentUpgradeableProxy.sol	0x164315...421764b6	Proxy for iGOLDx
TransparentUpgradeableProxy.sol	0x47566a...3c5f3698	Proxy for iHBTC
TransparentUpgradeableProxy.sol	0xA3068A...cf3ca7DE	Proxy for iLINK
TransparentUpgradeableProxy.sol	0x591595...6638f34B	Proxy for iMEUX
TransparentUpgradeableProxy.sol	0x039E7E...385c24bc	Proxy for iMKR
TransparentUpgradeableProxy.sol	0xd1254d...48483cEE	Proxy for iMUSX
TransparentUpgradeableProxy.sol	0xfa2e83...38B298Dd	Proxy for iMxBTC
TransparentUpgradeableProxy.sol	0x028DB7...6aB0FaA6	Proxy for iMxETH
TransparentUpgradeableProxy.sol	0x6E6a68...20ea2DBe	Proxy for iTUSD

File name	Contract deployed	Comment
TransparentUpgradeableProxy.sol	0xbeC9A8...9d396320	Proxy for iUNI
TransparentUpgradeableProxy.sol	0x2f956b...83BF0f45	Proxy for iUSDC
TransparentUpgradeableProxy.sol	0x1180c1...08Ab5354	Proxy for iUSDT
TransparentUpgradeableProxy.sol	0x1AdC34...41979eb0	Proxy for iUSX
TransparentUpgradeableProxy.sol	0x5812fC...07f63c02	Proxy for iWBTC
TransparentUpgradeableProxy.sol	0x6D9Ce3...83cf4a50	Proxy for icbBTC
TransparentUpgradeableProxy.sol	0x33b5Ed...5f3f02AA	Proxy for irETH
TransparentUpgradeableProxy.sol	0x590552...769158b9	Proxy for irenFIL
TransparentUpgradeableProxy.sol	0x5f02fb...93F795ba	Proxy for isDAI
TransparentUpgradeableProxy.sol	0x041D2c...bB1d5820	Proxy for isUSX
TransparentUpgradeableProxy.sol	0xbfd291...cd4848b9	Proxy for iwstETH
TransparentUpgradeableProxy.sol	0x4013e6...4fA07B35	Proxy for ixBTC
TransparentUpgradeableProxy.sol	0x237C69...72e425d6	Proxy for ixETH
iTokenV2.sol	0x379418...33c00F68	Implementation

Arbitrum:mainnet

File name	Contract deployed	Comment
TimeLock.sol	0x1E96e9...0a61b93d	
TransparentUpgradeableProxy.sol	0x8E7e9e...421E5408	Proxy for ControllerV2.sol
ControllerV2.sol	0xa32F91...DeA33403	Implementation
ControllerV2ExtraExplicit.sol	0x9F924E...64668f68	
ControllerV2ExtraImplicit.sol	0x9ad68d...47C7078A	
TransparentUpgradeableProxy.sol	0xEe3383...495dcC15	Proxy for iETH
iETHV2.sol	0xE68A13...54688deC	Implementation
TransparentUpgradeableProxy.sol	0xF45e2a...d96786C3	Proxy for RewardDistributorSecondV3.sol
RewardDistributorSecondV3.sol	0x8C6Fff...cA3d0D25	Implementation
TransparentUpgradeableProxy.sol	0x7702dC...e9F1D725	Proxy for iAAVE
TransparentUpgradeableProxy.sol	0xD037c3...6EeB95EF	Proxy for iARB
TransparentUpgradeableProxy.sol	0x662da3...AB639C0D	Proxy for iCRV
TransparentUpgradeableProxy.sol	0xf69959...D612b628	Proxy for iDAI
TransparentUpgradeableProxy.sol	0xaEa8e2...E29C4a63	Proxy for iDF
TransparentUpgradeableProxy.sol	0x567554...Ea3B0B8B	Proxy for iEUX
TransparentUpgradeableProxy.sol	0xb3ab71...8E6a504e	Proxy for iFRAX

File name	Contract deployed	Comment
TransparentUpgradeableProxy.sol	0x013ee4...449802C8	Proxy for iLINK
TransparentUpgradeableProxy.sol	0x5BE49B...02322021	Proxy for iMEUX
TransparentUpgradeableProxy.sol	0xe8c85B...b2fEA56c	Proxy for iMUSX
TransparentUpgradeableProxy.sol	0x46Eca1...12FEb17A	Proxy for iUNI
TransparentUpgradeableProxy.sol	0x8dc331...f84d4aE0	Proxy for iUSDC
TransparentUpgradeableProxy.sol	0x70284c...90b3B578	Proxy for iUSDCn
TransparentUpgradeableProxy.sol	0xf52f07...d49692a9	Proxy for iUSDT
TransparentUpgradeableProxy.sol	0x0385F8...0256cBAA	Proxy for iUSX
TransparentUpgradeableProxy.sol	0xD3204E...37eE0aCC	Proxy for iWBTC
TransparentUpgradeableProxy.sol	0xFD7e2E...aD9F69Df	Proxy for irETH
TransparentUpgradeableProxy.sol	0xB276FB...a88CA0BC	Proxy for isUSX
TransparentUpgradeableProxy.sol	0xa8bAd6...4B899b23	Proxy for iwstETH
iTokenV2.sol	0x7f5bB6...6F718867	Implementation

Optimism:mainnet

File name	Contract deployed	Comment

File name	Contract deployed	Comment
TimeLock.sol	0x0D535c...1316BAfe	
TransparentUpgradeableProxy.sol	0xA300A8...42d8BCF4	Proxy for ControllerV2.sol
ControllerV2.sol	0xF42661...FD945555	Implementation
ControllerV2ExtraExplicit.sol	0x379418...33c00F68	
ControllerV2ExtraImplicit.sol	0xE6b50f...62a72bB9	
TransparentUpgradeableProxy.sol	0xA7A084...5dA7b3B4	Proxy for iETH
iETHV2.sol	0x5A3aFF...cC8B167D	Implementation
TransparentUpgradeableProxy.sol	0x870ac6...Db9b71a2	Proxy for RewardDistributorSecondV3.sol
RewardDistributorSecondV3.sol	0xEc71C5...10E904f8	Implementation
TransparentUpgradeableProxy.sol	0xD65a18...181288d5	Proxy for iAAVE
TransparentUpgradeableProxy.sol	0xED3c20...53Aff36f	Proxy for iCRV
TransparentUpgradeableProxy.sol	0x5bedE6...bF78564c	Proxy for iDAI
TransparentUpgradeableProxy.sol	0x683236...F74A6cE6	Proxy for iDF
TransparentUpgradeableProxy.sol	0xDd40BB...48B28Ece	Proxy for iLINK
TransparentUpgradeableProxy.sol	0x94a14B...4445876A	Proxy for iMUSX
TransparentUpgradeableProxy.sol	0x7702dC...e9F1D725	Proxy for iOP

File name	Contract deployed	Comment
TransparentUpgradeableProxy.sol	0xB34479...041A2cc2	Proxy for iUSDC
TransparentUpgradeableProxy.sol	0x0fD11B...b07b6c46	Proxy for iUSDCn
TransparentUpgradeableProxy.sol	0x5d05c1...1148FC44	Proxy for iUSDT
TransparentUpgradeableProxy.sol	0x7e7e1d...7544Ce43	Proxy for iUSX
TransparentUpgradeableProxy.sol	0x24d302...8b06eB27	Proxy for iWBTC
TransparentUpgradeableProxy.sol	0x107d86...6976F71e	Proxy for irETH
TransparentUpgradeableProxy.sol	0x1f144c...628e2Ed7	Proxy for isUSD
TransparentUpgradeableProxy.sol	0xc0BD38...5062309C	Proxy for isUSX
TransparentUpgradeableProxy.sol	0x4B3488...7fE09A16	Proxy for iwstETH
iTokenV2.sol	0x2dcCE9...7051f9Cb	Implementation

Polygon:mainnet

File name	Contract deployed	Comment
TimeLock.sol	0x1C4d5e...4f6ED1C9	
TransparentUpgradeableProxy.sol	0x52eaCd...7F025f37	Proxy for ControllerV2.sol
ControllerV2.sol	0x379418...33c00F68	Implementation
ControllerV2ExtraExplicit.sol	0xE6b50f...62a72bB9	

File name	Contract deployed	Comment
ControllerV2ExtraImplicit.sol	0xfAEA99...1e7b1128	
TransparentUpgradeableProxy.sol	0x47C19A...9eF4022F	Proxy for RewardDistributorSecondV3.sol
RewardDistributorSecondV3.sol	0xF42661...FD945555	Implementation
TransparentUpgradeableProxy.sol	0x38D0c4...471Cd6f9	Proxy for iAAVE
TransparentUpgradeableProxy.sol	0x7D86eE...aF691B68	Proxy for iCRV
TransparentUpgradeableProxy.sol	0xec85F7...1496d95b	Proxy for iDAI
TransparentUpgradeableProxy.sol	0xcB5D9b...86F939B2	Proxy for iDF
TransparentUpgradeableProxy.sol	0x159624...1531aD6d	Proxy for iEUX
TransparentUpgradeableProxy.sol	0x6A3fE5...A0678c74	Proxy for iMATIC
TransparentUpgradeableProxy.sol	0x5268b3...Fcb65234	Proxy for iUSDC
TransparentUpgradeableProxy.sol	0xb3ab71...8E6a504e	Proxy for iUSDT
TransparentUpgradeableProxy.sol	0xc171EB...7CA29882	Proxy for iUSX
TransparentUpgradeableProxy.sol	0x94a14B...4445876A	Proxy for iWBTC
TransparentUpgradeableProxy.sol	0x0c9261...70d81740	Proxy for iWETH
iTokenV2.sol	0xEc71C5...10E904f8	Implementation

File name	Contract deployed	Comment
iETHV2.sol	0x2dcCE9...7051f9Cb	Implementation

BSC:mainnet

File name	Contract deployed	Comment
TimeLock.sol	0x8C3984...7D156D7a	
TransparentUpgradeableProxy.sol	0x0b53E6...27E6dc0A	Proxy for ControllerV2.sol
ControllerV2.sol	0xF42661...FD945555	Implementation
ControllerV2ExtraExplicit.sol	0x379418...33c00F68	
ControllerV2ExtraImplicit.sol	0xE6b50f...62a72bB9	
TransparentUpgradeableProxy.sol	0x390bf3...a6F47669	Proxy for iETH
iETHV2.sol	0x5A3aFF...cC8B167D	Implementation
TransparentUpgradeableProxy.sol	0x6fC21a...bDeF0Ef4	Proxy for RewardDistributorSecondV3.sol
RewardDistributorSecondV3.sol	0xEc71C5...10E904f8	Implementation
TransparentUpgradeableProxy.sol	0xFc5Bb1...Cab68862	Proxy for iADA
TransparentUpgradeableProxy.sol	0x55012a...65009Ef7	Proxy for iATOM
TransparentUpgradeableProxy.sol	0x9747e2...64193c15	Proxy for iBCH
TransparentUpgradeableProxy.sol	0xd57E14...67aA4A93	Proxy for iBNB

File name	Contract deployed	Comment
TransparentUpgradeableProxy.sol	0x0b66A2...E400356e	Proxy for iBTC
TransparentUpgradeableProxy.sol	0x5511b6...78204a47	Proxy for iBUSD
TransparentUpgradeableProxy.sol	0xeFae8F...72194d73	Proxy for iCAKE
TransparentUpgradeableProxy.sol	0xAD5Ec1...C50e5492	Proxy for iDAI
TransparentUpgradeableProxy.sol	0xeC3FD5...59e12D08	Proxy for iDF
TransparentUpgradeableProxy.sol	0x9ab060...6192725b	Proxy for iDOT
TransparentUpgradeableProxy.sol	0x983A72...D3f2B1d8	Proxy for iEUX
TransparentUpgradeableProxy.sol	0xD739A5...E58Fb810	Proxy for iFIL
TransparentUpgradeableProxy.sol	0xc35ACA...55B2D670	Proxy for iGOLDx
TransparentUpgradeableProxy.sol	0x50E894...4c963087	Proxy for iLINK
TransparentUpgradeableProxy.sol	0xd957be...d1c1e6ac	Proxy for iLTC
TransparentUpgradeableProxy.sol	0xb22eF9...f51511Eb	Proxy for iMEUX
TransparentUpgradeableProxy.sol	0x36f4C3...EDdE0991	Proxy for iMUSX
TransparentUpgradeableProxy.sol	0x6E4242...8023b4e5	Proxy for iMxBTC
TransparentUpgradeableProxy.sol	0x6AC0a0...53C72346	Proxy for iMxETH

File name	Contract deployed	Comment
TransparentUpgradeableProxy.sol	0xee9099...84B8806b	Proxy for iUNI
TransparentUpgradeableProxy.sol	0xAF9c10...9Bfe005d	Proxy for iUSDC
TransparentUpgradeableProxy.sol	0x0BF8C7...0008fa0F	Proxy for iUSDT
TransparentUpgradeableProxy.sol	0x7B933e...C8518aBe	Proxy for iUSX
TransparentUpgradeableProxy.sol	0x6D64eF...d526Dd6d	Proxy for iXRP
TransparentUpgradeableProxy.sol	0x8be8cd...30c335AA	Proxy for iXTZ
TransparentUpgradeableProxy.sol	0xc0bd38...5062309c	Proxy for isUSX
TransparentUpgradeableProxy.sol	0x219B85...0F1d3d2E	Proxy for ixBTC
TransparentUpgradeableProxy.sol	0xF649E6...eadFE05d	Proxy for ixETH
iTokenV2.sol	0x2dcCE9...7051f9Cb	Implementation

Conflux

File name	Contract deployed	Comment
TimeLock.sol	0x3f9E89...bEcCb236	
TransparentUpgradeableProxy.sol	0xA377eC...271f6a56	Proxy for ControllerV2.sol
ControllerV2.sol	0x1e3c5C...1EAB2B58	Implementation
ControllerV2ExtraExplicit.sol	0xD9ae4E...aF232E2b	

File name	Contract deployed	Comment
ControllerV2ExtraImplicit.sol	0x09Ae43...48dB96cD	
TransparentUpgradeableProxy.sol	0x620e8E...950EbA24	Proxy for iETH
iETHV2.sol	0x0a8e63...686BDfA1	Implementation
TransparentUpgradeableProxy.sol	0x3482f3...C65288c0	Proxy for RewardDistributorSecondV3.sol
RewardDistributorSecondV3.sol	0x7a2F91...0CE224Dd	Implementation
TransparentUpgradeableProxy.sol	0x25CCd7...742d88bc	Proxy for iCFX
TransparentUpgradeableProxy.sol	0xb88DC5...11AF9897	Proxy for iUSDC
TransparentUpgradeableProxy.sol	0xC80aD4...6ef5EB5b	Proxy for iUSDT
TransparentUpgradeableProxy.sol	0x6f87b3...55db6358	Proxy for iUSX
TransparentUpgradeableProxy.sol	0xE08020...F5035d75	Proxy for iWBTC
iTokenV2.sol	0x79E333...ab31AAE1	Implementation

1.5 Summary of findings

Severity	# of Findings
Critical	2
High	5
Medium	7
Low	11

ID	Name	Severity	Status
C-1	Inflation attack on iToken	Critical	Fixed
C-2	A detached reward distributor can be drained	Critical	Fixed
H-1	The `ControllerV2` implementation can be destroyed by an attacker	High	Fixed
H-2	Manipulation of global daily limits on `TimeLockStrategy`	High	Acknowledged
H-3	Funds may freeze on the `TimeLock` if the beneficiary does not implement `claim()`	High	Fixed
H-4	Tokens in Segregated mode cannot be fully repaid by borrowers	High	Fixed
H-5	Seizing assets as collateral without entering the market may result in incorrect value calculation	High	Fixed
M-1	`_exitMarket` always returns `true`, even on error	Medium	Fixed
M-2	Lack of speed-up functionality in the `TimeLock`	Medium	Fixed

M-3	Potential desynchronization between asset transfer and agreement creation in the `TimeLock`	Medium	Acknowledged
M-4	The lack of support of fee on transfer tokens in `DefaultTimeLock`	Medium	Acknowledged
M-5	Changing the `rewardToken` during distribution in `RewardDistributor` is dangerous	Medium	Acknowledged
M-6	Vulnerabilities to rug pull scenarios	Medium	Acknowledged
M-7	Assets may be unexpectedly seized	Medium	Acknowledged
L-1	`isController` reports `true` on the implementation contract	Low	Acknowledged
L-2	`extraImplicit` and `extraExplicit` are declared twice	Low	Fixed
L-3	A redundant `market` parameter in `exitMarketFromiToken`	Low	Acknowledged
L-4	A misleading function name `unfreezeAllAgreements`	Low	Fixed
L-5	The lack of verification of `timeLock.controller` in `_setTimeLock` setter	Low	Fixed
L-6	Missing validations for non-zero `mintAmount`, `borrowAmount` and `repayAmount`	Low	Acknowledged
L-7	Permit logic doesn't follow the ERC-2612 specification	Low	Acknowledged
L-8	The Solidity version is not up to date	Low	Acknowledged
L-9	Unintended ETH `receive` in the Controller	Low	Fixed
L-10	Using OpenZeppelin `__disableInitializers` in ControllerV2ExtraBase	Low	Acknowledged
L-11	Using the OpenZeppelin `EnumerableSetUpgradeable.values()` function	Low	Acknowledged

1.6 Conclusion

The project encountered well-known issues such as inflation attack and proxy implementation self-destruction, which according to the developers were known to them and were supposed to be addressed through the correct deployment procedure. We recommend always resolving such issues through the code of smart contracts.

We also recommend enhancing the test coverage by better evaluating both positive and negative scenarios in the behavior of functions.

It is also important to remember that the user of the system can be not only an EOA (Externally Owned Account) but also a smart contract, which imposes certain limitations on the user's ability to interact with the system.

Dividing the Controller code into several smart contracts with non-trivial mutual calls complicates reading and analyzing the code. To enhance security, we recommend using simpler architectural solutions whenever possible.

During the audit, 2 critical, 5 high, 7 medium and 11 low severity issues have been discovered. All issues are confirmed by the developers and fixed or acknowledged.

2. FINDINGS REPORT

2.1 Critical

C-1	Inflation attack on iToken
Severity	Critical
Status	Fixed in ebeee963

Description

Until `iToken` has sufficient `totalSupply`, an attacker can manipulate the `underlying`/`iToken` exchange rate by directly transferring the underlying asset to the `iToken` smart contract. This leads to rounding issues in `mint` and `redeemUnderlying` causing a user to lose some amount of their underlying assets.

Due to the possibility of permanent loss of user assets, such issues have a critical severity rating.

Related code:

- rounding issues on mint
  Base.sol#L199
- rounding issues on redeem underlying in iToken for native token
  iETH.sol#L140
- rounding issues on redeem underlying in iToken for ERC20 iToken.sol#L126

Recommendation

Although this issue can be hotfixed through accurate deployment procedures and configuration settings, we recommend fixing it at the smart contract code level either by preventing the iToken from having a nonzero but small `totalSupply` or by ensuring accurate accounting of the underlying asset in the smart contract.

C-2	A detached reward distributor can be drained
Severity	Critical
Status	Fixed in 490a30f5

Description

If admin decided to change the current reward distribution logic and set new one by using the Controller.sol#L548 function,
the prev version is supposed to distribute rewards for the prev period.
After detaching the reward distribution contract from the controller, transfers don't track by the controller any more and by abusing this issue an attacker can drain rewards from the old distributor by using cycles charge balance then claim from different accounts or a flashloan attack.

Recommendation

We recommend following one of the two ways:

- allow tracking transfers by a few distributors at the same time,
- don't change the distributor address and use migration.

2.2 High

H-1	The `ControllerV2` implementation can be destroyed by an attacker
Severity	High
Status	Fixed in bf28390f

Description

The `ControllerV2` implementation code is vulnerable to a direct call of `initialize`. Since `initialize` executes `delegatecall` to an arbitrary address, an attacker can destroy the Controller's implementation contract, thus freezing the entire system until manual intervention by the proxy administrator occurs. This is accordingly rated as high in severity.

Related code - `delegatecall` to the arbitrary address: ControllerV2.sol#L57

Recommendation

Although this vulnerability can be hotfixed through an accurate deployment process, we recommend addressing it at the smart contract code level by preventing direct calls to `initialize` against the implementation address.

H-2	Manipulation of global daily limits on `TimeLockStrategy`
Severity	High
Status	Acknowledged

Description

The global daily limits implemented in the audited code are susceptible to manipulation by an attacker, leading to inconvenience for legitimate users due to the time lock on any outgoing transfers. Since the system will remain in an undesired state until the smart contract owner intervenes, this issue is rated as high in severity.

Related code - procedure for accumulating daily statistics:
TimeLockStrategy.sol#L166

Recommendation

We recommend reworking the global limits to prevent manipulation.

Client's commentary

> We are aware of this, and working on a more sophisticated strategy to decide the delay of a outgoing transfer.

H-3	Funds may freeze on the `TimeLock` if the beneficiary does not implement `claim()`
Severity	High
Status	Fixed in 8ad82e8e

Description

Assets from the `TimeLock` can only be claimed by their respective beneficiaries via calling the `claim` function. However, if the beneficiary is an immutable smart contract with no ability to invoke `claim` against the `TimeLock`, the locked assets become inaccessible to the beneficiary. Given that some accounts will be unable to access their assets until the manual intervention of the smart contract owner, this issue is rated as high in severity.

Related code - procedure of agreement execution:
DefaultTimeLock.sol#L81

Recommendation

We recommend allowing any account to invoke the `claim`.

H-4	Tokens in Segregated mode cannot be fully repaid by borrowers
Severity	High
Status	Fixed in 820d9182

Description

Tokens that have the Segregated mode activated possess a designated `MarketV2.currentDebt` value. This value is prevented from surpassing the `debtCeiling` through borrow functions. Notably, the `ControllerV2ExtraExplicit.afterRepayBorrow` function employs the `SafeMath.sub` function to subtract the amount of repaid underlying assets from the `currentDebt` value. This function is designed to revert any underflow errors. However, the `currentDebt` value does not consider that the debt is increasing over time with `InterestRateModel`, associated with the `iToken`. Consequently, the repaid amount always exceeds the borrowed sum, causing borrowers unable to fully repay their debt until the contract's owner updates the `ControllerV2ExtraExplicit` implementation.

This issue is labeled as `high`, since it imposes the potential to temporarily block specific `repayBorrow` transactions.

Related code - `beforeBorrow` for Segregated mode: ControllerV2ExtraExplicit.sol#L200

Recommendation

We recommend reseting the `currentDebt` value to `zero` in cases where `currentDebt` is less than `repayAmount`.

H-5	Seizing assets as collateral without entering the market may result in incorrect value calculation
Severity	High
Status	Fixed in fa5cfaf1

Description

During liquidation, collateral may be seized even if the borrower has not entered the market with it. Sanity checks regarding the price oracle status for the seized asset will be skipped if the market has not been entered for this asset.

This issue is labeled as `high` since an outdated or inaccurate `iTokenCollateral` price could result in either excessive or insufficient payments to the liquidator.

Related code:

- the `liquidateCalculateSeizeTokensV2` function: ControllerV2ExtraImplicit.sol#L477
- `_liquidateBorrowInternal` iTokenV2.sol#L76
- `beforeLiquidateBorrow` ControllerV2.sol#L346

Recommendation

We recommend prohibiting the seizure of assets that are not explicitly listed by the borrower as allowed collateral through `enterMarket`.

2.3 Medium

M-1	`_exitMarket` always returns `true`, even on error
Severity	Medium
Status	Fixed in 6315b9a2

Description

The `_exitMarket` function, as per its specification, is designed to return `false` if the market isn't listed or not entered. However, in its current implementation, the function always returns `true`, leading to inconsistency between the expected and actual outcomes.

Related code:

- function returns `true` even if the token is not listed Controller.sol#L1401
- function returns `true` even if the market is not entered Controller.sol#L1406

Recommendation

We recommend adjusting the `_exitMarket` function to return values in accordance with the expectations of both users and developers as well as the specification.

M-2	Lack of speed-up functionality in the `TimeLock`
Severity	Medium
Status	Fixed in 1df8a1da

Description

Once created, an agreement in the `TimeLock` enforces a delay until the expiration time specified during the agreement's creation. If the delays are unintentionally long, the only remedy is to replace the `TimeLock` implementation.

Related code - procedure of agreement execution: DefaultTimeLock.sol#L83

Recommendation

We recommend implementing speed-up functionality in the `TimeLock` to address unintentionally prolonged delays.

M-3	Potential desynchronization between asset transfer and agreement creation in the `TimeLock`
Severity	Medium
Status	Acknowledged

Description

In the audited code, asset transfer and agreement creation are treated as two separate processes.

- Assets can be transferred to the `TimeLock` without creating an agreement, leading to them being frozen.
- An agreement can be created without transferring assets and may be satisfied using assets intended for other agreements, rendering those agreements unsatisfiable.

Related code - agreement creation: DefaultTimeLock.sol#L47

Recommendation

Although the asset transfer and the agreement creation are currently synchronized (outside of the `TimeLock` smart contract), we recommend synchronizing them within the `TimeLock` smart contract itself to maintain the `TimeLock` state consistency.

Client's commentary

> Such solution is chosen as it simplifies the logic of `iToken`'s outgoing transfer and does not require additional `approve` to `TimeLock`.

M-4	The lack of support of fee on transfer tokens in `DefaultTimeLock`
Severity	Medium
Status	Acknowledged

Description

The agreement creation precedes the token transfer. If a fee on transfer tokens is used, then the amount of tokens transferred may be reduced (due to transfer fees) and become less than the amount specified in the agreement for claiming.

This issue is labeled as `medium` since the resulted inconcistency can block the `claim` function until the `balance` of the timelock surpasses the quantity of tokens noted in the agreement.

Related code - agreement creation: DefaultTimeLock.sol#L47

Recommendation

We recommend reworking the `TimeLock` architecture to pull assets by `transferFrom` in the `createAgreement`. Additionally, it will help addressing previous finding.

Client's commentary

> We are aware of it and will carefully choose assets onboarding.

M-5	Changing the `rewardToken` during distribution in `RewardDistributor` is dangerous
Severity	Medium
Status	Acknowledged

Description

Alterations to the `rewardToken` in the middle of distribution, especially without verifying the congruence of `decimals` between the previous and the new token and ensuring price consistency, can lead to potential risks of excessive or insufficient rewards to distribution recipients.

Related code - procedure of updating the reward token: RewardDistributorV3.sol#L105

Recommendation

We recommend disabling the `_setRewardToken` function if the current `rewardToken` is a non-zero address.

Client's commentary

> `_setRewardToken` will normally only be set once, we prefer to keep some flexibility here.

M-6	Vulnerabilities to rug pull scenarios
Severity	Medium
Status	Acknowledged

Description

The contracts are `Ownable` with a possibility to change the contracts implementation to arbitrary code. Also, some contracts have functions to retreive the `ERC-20` tokens by the owner (e.g. `RewardDistributorV3.rescueTokens`, `iToken._withdrawReserves`).

Recommendation

To minimize the risk of a rug pull, we recommend utilizing the MultiSig and TimeLock techniques as the owner to ensure that no single entity has unilateral control. In the long run, consider transitioning to a DAO for governance functions.

Client's commentary

> Currently the ownership is ultimately controlled by a MultiSig and the governance process can be found on https://snapshot.org/#/dforcenet.eth.

M-7	Assets may be unexpectedly seized
Severity	Medium
Status	Acknowledged

Description

During liquidation, collateral may be seized even if the borrower has not entered the market with it. Such behavior is likely unexpected for the borrower.

Related code:

- the `liquidateCalculateSeizeTokensV2` function: ControllerV2ExtraImplicit.sol#L477
- `_liquidateBorrowInternal` iTokenV2.sol#L76
- `beforeLiquidateBorrow` ControllerV2.sol#L346

Recommendation

We recommend prohibiting the seizure of assets that are not explicitly listed by the borrower as allowed collateral through `enterMarket`.

Client's commentary

> It is a *feature* to ensure the protocol's solvency.

2.4 Low

L-1	`isController` reports `true` on the implementation contract
Severity	Low
Status	Acknowledged

Description

The `isController` view function is designed to prevent the accidental specification of an incorrect smart contract address as the controller address. However, the address of the `Controller` implementation incorrectly returns `true`, even though the valid controller address is intended to be a proxy address, not the implementation address.

Related code - `isController` view function: Controller.sol#L60

Recommendation

To enhance sanity checks, we recommend ensuring `isController` returns `false` when called against the implementation address.

Client's commentary

> We prefer to keep the proxy a pure proxy, and in some scenarios (mostly test cases), the controller are non-proxied.

L-2	`extraImplicit` and `extraExplicit` are declared twice
Severity	Low
Status	Fixed in 27e7df6a

Description

The storage variables `extraImplicit` and `extraExplicit` are declared twice in the code:

- ControllerV2.sol#L39-L43
- ControllerStorage.sol#L221-L225

  This could potentially lead to unexpected behavior.

Recommendation

We recommend removing the redundant declaration in `ControllerV2.sol`.

L-3	A redundant `market` parameter in `exitMarketFromiToken`
Severity	Low
Status	Acknowledged

Description

The `exitMarketFromiToken` function is designed to let the `iToken` request an exit from the market for a specified account using the given `iToken`. However, by providing a market parameter different from `address(this)`, the iToken can be permitted to exit from a market other than its own.

Related code - exit from an arbitrary market: ControllerV2.sol#L512

Recommendation

We recommend eliminating the `market` parameter and utilizing `msg.sender` as a secure substitute.

Client's commentary

> `exitMarketFromiToken` is the conterpart of `enterMarketFromiToken`, which can be called from a `iTokenB` to collateralize `iTokenC` by a modified version(iMSDMiniPool.sol#L200).
> We prefer to keep the interface consistant.

L-4	A misleading function name `unfreezeAllAgreements`
Severity	Low
Status	Fixed in dd99b2ab

Description

Despite its name, the `unfreezeAllAgreements` function does not actually unfreeze all agreements. It merely removes the global freeze flag that applies to all agreements, but an agreement will remain frozen if it was previously frozen by `freezeAgreements`.

Related code - procedure of agreement execution: DefaultTimeLock.sol#L86

Recommendation

We recommend changing the name of the `unfreezeAllAgreements` function to one that is more indicative of its actual functionality.

L-5	The lack of verification of `timeLock.controller` in `_setTimeLock` setter
Severity	Low
Status	Fixed in 31d2a460

Description

The `_setTimeLock` setter does not perform verification whether `timeLock.controller` is equivalent to `address(this)` or not. This may lead to all the `transferOut` function for `iTokens` becoming inaccessible.

Related code - `_setTimeLock`: ControllerV2ExtraImplicit.sol#L57

Recommendation

We recommend ensuring the equivalence of `timeLock.controller` and `address(this)` within the `_setTimeLock` function.

L-6	Missing validations for non-zero `mintAmount`, `borrowAmount` and `repayAmount`
Severity	Low
Status	Acknowledged

Description

In the current codebase, validations ensuring that `mintAmount`, `borrowAmount` and `repayAmount` are greater than zero are absent in the `iToken.mint`,`iToken.borrow`, `iToken.repayBorrow` functions respectively.

These missing checks can lead to unintended consequences, such as misleading event emissions or and registering empty collateral or borrow assets to users.

Related code:

- `mintInternal` Base.sol#L179
- `borrowInternal` Base.sol#L261
- `repayInternal` Base.sol#L299

Recommendation

We recommend inserting validation checks ensuring that the amounts are greater than zero to the following functions: `Base.borrowInternal`, `Base.repayInternal`, `Base.mintInternal`.

Client's commentary

> Prefer to keep it as it is.

L-7	Permit logic doesn't follow the ERC-2612 specification
Severity	Low
Status	Acknowledged

Description

The `iToken` uses a non-standard way to implement `permit`. It may cause compatibility issues when used in a third-party project.

Related code - implementation of `permit` in the `iToken`: Base.sol#L524

Recommendation

We recommend using the way that OpenZeppelin recommends (ERC20Permit.sol).

Client's commentary

> Prefer to keep it as it is.

L-8	The Solidity version is not up to date
Severity	Low
Status	Acknowledged

Description

The modern major version of the Solidity compiler is 0.8, but most of the codebase uses version 0.6.12.

Recommendation

We recommend using the up to date Solidity version.

Client's commentary

> Prefer to keep it as it is.

L-9	Unintended ETH `receive` in the Controller
Severity	Low
Status	Fixed in 6845977f

Description

At line ControllerV2.sol#L167
there is a receiver declared that isn't used anywhere.

Recommendation

We recommend removing the unused `receive()` function to prevent sending ETH to the contract.

L-10	Using OpenZeppelin `__disableInitializers` in ControllerV2ExtraBase
Severity	Low
Status	Acknowledged

Description

To avoid the ability to directly call the initialize() function at the implementation contract address, the constructor currently calls the initialize() function.

```
constructor() public {
    __initialize();
}

function __initialize() internal initializer {
    __Ownable_init();
}
```

OpenZeppelin provides a special function intended to disable initializers from the constructor.

Recommendation

We recommend using the OpenZeppelin `_disableInitializers` function.

Client's commentary

> We've chosen to maintain our current usage due to the absence of this implementation in OpenZeppelin version 3.3.0, ensuring consistency within the project.

L-11	Using the OpenZeppelin `EnumerableSetUpgradeable.values()` function
Severity	Low
Status	Acknowledged

Description

Currently, a loop is used to retrieve the values of the `EnumerableSetUpgradeable`. However, there is a special function intended to retrieve the values of the `EnumerableSetUpgradeable`.

Recommendation

We recommend using the `values` function to retreive the values of the `EnumerableSetUpgradeable`.

Client's commentary

> We've chosen to maintain our current usage due to the absence of this implementation in OpenZeppelin version 3.3.0, ensuring consistency within the project.

3. ABOUT MIXBYTES

MixBytes is a team of blockchain developers, auditors and analysts keen on decentralized systems. We build opensource solutions, smart contracts and blockchain protocols, perform security audits, work on benchmarking and software testing solutions, do research and tech consultancy.

Contacts

https://github.com/mixbytes/audits_public

https://mixbytes.io/

hello@mixbytes.io

https://twitter.com/mixbytes