DIVERGENCE PROTOCOL SECURITY AUDIT REPORT

December 28, 2023

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

Divergence Protocol is an on-chain derivatives exchange. Initially, the protocol focuses on creating an AMM-based marketplace trading synthetic derivative tokens with a binary pay-off structure. It allows the decentralized creation of so-called `Battles` where `Spear` and `Shield` derivatives are traded and settled based on oracle-supplied price feeds.

1.4 Project Dashboard

Project Summary

Title	Description
Client	Divergence Protocol
Project name	Divergence Protocol
Timeline	29.05.2023 - 26.12.2023
Number of Auditors	3

Project Log

Date	Commit Hash	Note
29.05.2023	e5286f94a7ccb9d6279fae51ea66a8833672628a	initial commit for the audit
03.07.2023	cc987fedf3783d3498270afb09050afaed4f8c0b	commit with fixes for the reaudit
04.10.2023	29a0ccb5fc7ac838bd44c75c0afc398b84be267a	commit with additional functionality
16.10.2023	1c0b9317e38bdddb541259d04d0e4a37a6f6ea88	commit with the fixes
20.11.2023	f2df3c694e41e20151d77dc9aaf32296f63c5ef7	additional fixes
18.12.2023	f9378a8ba9f4d9d31ba94d08b20be821e16caf4b	commit with the final fixes

Project Scope

The audit covered the following files:

File name	Link
src/core/Arena.sol	Arena.sol
src/core/Battle.sol	Battle.sol
src/core/Oracle.sol	Oracle.sol
src/core/OracleCustom.sol	OracleCustom.sol
src/core/utils.sol	utils.sol
src/core/libs/DiverSqrtPriceMath.sol	DiverSqrtPriceMath.sol
src/core/libs/Position.sol	Position.sol
src/core/libs/TickMath.sol	TickMath.sol
src/core/libs/Tick.sol	Tick.sol
src/core/libs/TradeMath.sol	TradeMath.sol
src/core/token/SToken.sol	SToken.sol
src/periphery/Manager.sol	Manager.sol
src/periphery/base/BattleInitializer.sol	BattleInitializer.sol
src/periphery/base/LiquidityManagement.sol	LiquidityManagement.sol
src/periphery/base/PeripheryImmutableState.sol	PeripheryImmutableState.sol
src/periphery/base/PeripheryPayments.sol	PeripheryPayments.sol
src/periphery/lens/Quoter.sol	Quoter.sol
src/periphery/libs/CallbackValidation.sol	CallbackValidation.sol
src/periphery/libs/DiverLiquidityAmounts.sol	DiverLiquidityAmounts.sol

Deployments

Contract	Address	Comment
OracleCustom	0x1aBAE7617f6Ae462412788E5031dA3646F188607	
Oracle	0x07C8C1e84C3814bfE2870Dfed24635Ef715aCB7c	
Quoter	0x2791386ecE62d8Ec3D3b223871be192c6BD5D5D5	
Arena	0xA0D812cAe2376b90951192319477eF5Fe3Ac56D5	
Battle	0xB9A1B8dF279E20Ff7dC7A32EE15d3eF2d5E45F62	
Manager	0x55A14661d94C2cE307Ab918bb9564545282C2454	

1.5 Summary of findings

Severity	# of Findings
Critical	4
High	4
Medium	5
Low	5

ID	Name	Severity	Status
C-1	Debug code in `getPriceByExternal`	Critical	Fixed
C-2	Battle fishing attack	Critical	Fixed
C-3	Multiple redemptions of position are possible	Critical	Fixed
C-4	Incorrect price calculation on more than two phases	Critical	Fixed
H-1	The `startSqrtPriceX96` manipulation	High	Acknowledged
H-2	Empty trades	High	Fixed
H-3	Price manipulation by calling `updatePhase` with different symbols	High	Fixed
H-4	Price manipulation by calling `updatePhase` prematurely	High	Fixed
M-1	The owner can manipulate oracles	Medium	Fixed
M-2	Reserve oracle	Medium	Acknowledged

M-3	Public functions for withdrawn assets in the `PeripheryPayments` contract	Medium	Acknowledged
M-4	Lack of incentivization of the battle settlement process	Medium	Acknowledged
M-5	The `redeemObligation` will always be reverted	Medium	Fixed
L-1	Spelling mistakes	Low	Fixed
L-2	Only manager modifier	Low	Fixed
L-3	Using a lock modifier	Low	Acknowledged
L-4	The use of immutable variables	Low	Fixed
L-5	Magic numbers	Low	Acknowledged

1.6 Conclusion

The project contains a complex mathematical foundation, which appears to be well-developed, and no findings have been discovered in it.

The majority of errors, including 3 out of 4 critical-level errors, seem to be the result of under-preparation for the audit. To ensure safe code, we recommend that developers allocate sufficient resources to maintain code quality and testing, regardless of whether this code will undergo an independent audit process or not.

The discovered errors (4 critical, 4 high, 5 medium, 5 low) have been confirmed by the developer and either corrected or acknowledged as not having a significant impact on the security of the project. The list of errors is below.

2.FINDINGS REPORT

2.1 Critical

C-1	Debug code in `getPriceByExternal`
Severity	Critical
Status	Fixed in cc987fed

Description

The last instruction of the `getPriceByExternal` function will always return the same price (30_000e18).
Oracle.sol#L43
It leads to the incorrect settles of battles. An attacker can use this code issue for getting profit from bets.

Recommendation

We recommend removing the `return (30_000e18, 0)` instruction from the function.

C-2	Battle fishing attack
Severity	Critical
Status	Fixed in cc987fed

Description

The `Arena` smart contract contains a public Arena.sol#L67 function that creates a new battle without initializing. This function is used in the `BattleInitializer`'s function BattleInitializer.sol#L14. It means that an attacker can create a new battle and binds it to a fake manager behind the scene. Using the manager priviledge, the fake manager contract can mint spear and shield tokens and put them to the pool. A user will trade these tokens for collateral. At the end of the battle the attacker could withdraw assets using the Battle.sol#L271 function of the battle.

Recommendation

We recommend improving the access right model to disallow attackers to gain privileged access.

C-3	Multiple redemptions of position are possible
Severity	Critical
Status	Fixed in 0cbf8712

Description

The `redeemObligation` function does not change the state of the position to `ObligationRedeemed`. This allows an attacker to call it multiple times, potentially gaining an unlimited amout of the collateral token. Currently, this issue is unexploitable because of issue M.5 (described below), however, it becomes relevant if issue M.5 is fixed.

Recommendation

We recommend changing the state of the position after redemption of the obligations to prevent multiple redemptions.

C-4	Incorrect price calculation on more than two phases
Severity	Critical
Status	Fixed in 1c0b9317

Description

In the `_getPrice` function, probably due to a mistype, the cycle is not exited after the determination of the final price. This may lead to several executions of the `price *= decimalDiff` code wich renders the price significantly higher than it should be.

As this issue may lead to an incorrect battle resolution, we assign the CRITICAL severity rating to it.

Related code - the `_getPrice` function: Oracle.sol#L66

Recommendation

We recommend fixing the algorithm of price calculation.

2.2 High

H-1	The `startSqrtPriceX96` manipulation
Severity	High
Status	Acknowledged

Description

The starting price Battle.sol#L102 is an arbitrary, user-defined value. For the given battle key, an attacker can specify their unfair price to grief other users or even to make profits from unfair trade conditions. Such attacks can't be avoided without redeploying the system's smart-contract.

Recommendation

We recommend including `startSqrtPriceX96` into the battle identifier (battle key).

Client's commentary

> At a protocol level, we intend to avoid introducing systematic bias in startSqrtPriceX96, which is unique to each Battle. Generally the opening price of an options market is determined by a collective of market participants. If startSqrtPriceX96 is unfair, it typically suggests the first liquidity position attempts to sell calls (or puts) at an unfair price. In this case buyers can wait for puts (or calls) liquidity to show up at a "fair" price, before a trade. As a protocol, we do not participate in the options market. If we arbitrarily set startSqrtPriceX96, we risk setting unfair prices to systematically grief our users.

H-2	Empty trades
Severity	High
Status	Fixed in 1c0b9317

Description

The trade function allows you to specify a price limit, upon reaching which the cycle will be exited without performing any movement of tokens at line
Battle.sol#L286,
but with a change of the current price at line
Battle.sol#L359.

An attacker creates a contract that calls the trade function of the Battle contract with a price limit. To get passed the balance check, the attacker sends one token to the contract inside tradeCallback.

This ability to perform empty exchanges in empty areas of liquidity creates the possibility of price manipulation. This capability can be used by an attacker to attack liquidity providers in order to block the addition of liquidity.

LiquidityManagement.sol#L43

To resolve this, it is necessary to resort to non-standard actions, for example, adding liquidity over the entire tick interval.

This error is marked as HIGH as the contract is blocked by the attacker.

Recommendation

We recommend following one of the next ways:

1. Revert empty trades
2. Add function initAddLIquidity that moves price to the right place and then call the addLiquidity function

H-3	Price manipulation by calling `updatePhase` with different symbols
Severity	High
Status	Fixed in 1c0b9317

Description

In the audited commit `29a0ccb5fc7ac838bd44c75c0afc398b84be267a`, the `endRoundId` value is not dependent on the asset symbol. By calling the `updatePhase` function with a different `symbol` argument, an attacker may manipulate the `endRoundId` values. This could violate the correctness of the price calculation algorithm, leading to incorrect battle income.

Although this is a critical flaw, it is only exploitable during a 1-hour period after a Chainlink phase has been changed. Given its critical impact but low likelihood, this issue is assigned a HIGH severity rating.

Related code - the declaration of the `endRoundId`: Oracle.sol#L17

Recommendation

We recommend reworking the entire algorithm of interaction with Chainlink to render it more tolerant to manipulations.

H-4	Price manipulation by calling `updatePhase` prematurely
Severity	High
Status	Fixed in 1c0b9317

Description

When `updatePhase` is called before the current phase has ended, it records an incorrect `endRoundId` value. Additionally, if `updatePhase` is not called after the phase has ended, the price calculation will be compromised in a manner similar to issue High.3.

This issue is even less exploitable than the previous one, but the impact is still critical, hence we assign it a HIGH severity rating.

Related code - the `updatePhase` function: Oracle.sol#L113

Recommendation

We recommend reworking the entire algorithm of interaction with Chainlink to render it more tolerant to manipulations.

2.3 Medium

M-1	The owner can manipulate oracles
Severity	Medium
Status	Fixed in cc987fed

Description

The `Oracle` contract has a function Oracle.sol#L24 that set an external oracle address for symbols. The owner is able to set a fake external oracle just before the battle end. As a result, it will lead to the incorrect battle settlement.

Recommendation

We recommend locking the possibility to change the oracle address for an already created battle.

Client's commentary

> The 'Oracle' contract function [`setExternalOracle`] sets external oracles for all on-going and new battles, in case the external oracle address has to change for a particular underlying price feed. While we accept your recommendation, it must be noted that, as a consequence, an already created battle cannot adjust its external oracle address, even if it is no longer functional.

M-2	Reserve oracle
Severity	Medium
Status	Acknowledged

Description

In case of some instability with Chainlink oracle gaps in a price history are probably present. These gaps cause the `settle` function call to revert.

Recommendation

We recommend adding some reserve price source for the `Oracle` contract.

Client's commentary

> We have an agreement with Chainlink, which will set a heartbeat at our specified time of settlement. Thus far, alternative price sources do not carry the necessary security guarantee of Chainlink's decentralized oracle network. Furthermore, alternative sources will give prices of differing specifications, which may introduce additional attack surfaces.
>
> We think there are future possibilities of collaborating with Chainlink to address the timeliness of price retrieval. A number of alternative solutions are under development at Chainlink, and possible integrations are not included in the current scope.

M-3	Public functions for withdrawn assets in the `PeripheryPayments` contract
Severity	Medium
Status	Acknowledged

Description

The PeripheryPayments.sol#L31 and PeripheryPayments.sol#L42 functions allow anyone to withdraw funds from the `Manager` contract. There are no funds on the contract by design but if someone makes a wrong transfer to the contract by mistake, an attacker can steal it. Also, there is a risk of inheritance of this contract in the future in other cases where funds will be kept on a new contract.

Recommendation

We recommend adding access control to the withdraw functions.

Client's commentary

> This is an issue that is not unique to our protocol implementation and is not induced by the working logic of our contracts. In this case additional access control will have to be granted to the owner, which is not a mandate for protocol functionality.

M-4	Lack of incentivization of the battle settlement process
Severity	Medium
Status	Acknowledged

Description

There is no specific incentivization of the Battle.sol#L424 function. Though most of the users are interested in battle finalization, but in order to save gas each user would prefer to avoid making the settlement by themselves. It may make sense for some networks where gas is expensive enough. Consecutively, it is expected behavior that users wouldn't make battle settlements, and the project owner will make it in a centralized manner even if it is unprofitable for them. It causes some centralization and an open attack vector for the griefing attack by spamming the project by battles that the project owner is enforced to finalize even if it is unprofitable for them.

Recommendation

We recommend introducing some incentivization of the battle settlement. It can be combined with spam protection.

Client's commentary

> Certain types of settlement incentivization may encourage spam. This is because we do not set minimums for position or trade size for a pool, and the pool's collateral token can be unrestricted. We expect rational participants will be incentivized as a stakeholder of positions within a pool. If left unsettled, one's own positions are at risk of not being settled in time, making it impossible to withdraw. So long as the need to immediately withdraw after settlement, the gas price is unlikely to dissuade users from calling [`settle()`].

M-5	The `redeemObligation` will always be reverted
Severity	Medium
Status	Fixed in 0cbf8712

Description

`redeemObligation` is designed to be called when the state is `Outcome.ONGOING`. However, it utilizes Manager.sol#L204 which will invariably be reverted under such conditions.

Recommendation

It is recommended to either modify `Battle.withdrawObligation` or use an alternative function instead in order to enable the withdrawal of obligations during the `Outcome.ONGOING` state.

2.4 Low

L-1	Spelling mistakes
Severity	Low
Status	Fixed in cc987fed

Description

passedƒ -> passed
Battle.sol#L163

Recommendation

We recommended fixing spelling mistakes.

L-2	Only manager modifier
Severity	Low
Status	Fixed in cc987fed

Description

There are a lot of similar checks for requirement that the calling address is a manager of the following kind:

```
    if (msg.sender != manager) {
        revert Errors.CallerNotManager();
    }
```

Code duplication makes the code base less clear and more voluminous.

Recommendation

We recommended adding `_requireOnlyManager` that checks that `msg.sender` is a manager.

L-3	Using a lock modifier
Severity	Low
Status	Acknowledged

Description

The `Battle` contract contains a [`lock`]
(Battle.sol#L78) modifier that prevents from reentrancy attacks but in
Battle.sol#L287 there are two `slot0.unlocked` modifications at the beginning
Battle.sol#L301
and at the end
Battle.sol#L420.

Recommendation

It is recommended to use a `lock` modifier instead of direct `slot0.unlocked` modifications and checks.

L-4	The use of immutable variables
Severity	Low
Status	Fixed in cc987fed

Description

The following variables can be marked as immutable to reduce gas consumption:
Arena.sol#L22
Arena.sol#L24

Recommendation

We recommend using immutable variables instead of storage variables.

L-5	Magic numbers
Severity	Low
Status	Acknowledged

Description

There are two numbers in this line
Arena.sol#L85.
The use of constants or Solidity time span build in keywords makes the codebase more understandable.

Recommendation

We recommend using constants or the Solidity time span build in keywords.

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