OPIUM NETWORK PROTOCOL V1 SMART CONTRACT AUDIT

January 30, 2021

MixBytes()

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 Opium Network. 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 PROJECT OVERVIEW

The Opium protocol is a universal protocol to create, settle and trade virtually all derivatives and financial instruments in a professional and trustless way. It allows anyone to build custom exchange-traded products on top of the Ethereum blockchain. Once created, they can be traded freely via a network of relayers and will be priced according to supply and demand.

1.3 SECURITY ASSESSMENT METHODOLOGY

At least 2 auditors are involved in the work on the audit who check the provided source code independently of each other in accordance with the methodology described below:

01 "Blind" audit includes:
> Manual code study
> "Reverse" research and study of the architecture of the code based on the source code only

Stage goal:
Building an independent view of the project's architecture
Finding logical flaws

02 Checking the code against the checklist of known vulnerabilities includes:
> Manual code check for vulnerabilities from the company's internal checklist
> The company's checklist is constantly updated based on the analysis of hacks, research and audit of the clients' code

Stage goal:
Eliminate typical vulnerabilities (e.g. reentrancy, gas limit, flashloan attacks, etc.)

03 Checking the logic, architecture of the security model for compliance with the desired model, which includes:
> Detailed study of the project documentation
> Examining contracts tests
> Examining comments in code
> Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit

Stage goal:
Detection of inconsistencies with the desired model

04 Consolidation of the reports from all auditors into one common interim report document
> Cross check: each auditor reviews the reports of the others
> Discussion of the found issues by the auditors
> Formation of a general (merged) report

Stage goal:
Re-check all the problems for relevance and correctness of the threat level
Provide the client with an interim report

05 Bug fixing & re-check.
> Client fixes or comments on every issue
> Upon completion of the bug fixing, the auditors double-check each fix and set the statuses with a link to the fix

Stage goal:
Preparation of the final code version with all the fixes

06 Preparation of the final audit report and delivery to the customer.

Findings discovered during the audit are classified as follows:

FINDINGS SEVERITY BREAKDOWN

Level	Description	Required action
Critical	Bugs leading to assets theft, fund access locking, or any other loss funds to be transferred to any party	Immediate action to fix issue
Major	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.	Implement fix as soon as possible
Warning	Bugs that can break the intended contract logic or expose it to DoS attacks	Take into consideration and implement fix in certain period
Comment	Other issues and recommendations reported to/acknowledged by the team	Take into consideration

Based on the feedback received from the Customer's team 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 project team is aware of this finding. Recommendations for this finding are planned to be resolved in the future. This finding does not affect the overall safety of the project.
No issue	Finding does not affect the overall safety of the project and does not violate the logic of its work.

1.4 EXECUTIVE SUMMARY

Audited scope includes main Opium protocol implementation. Implementation can be logically separated to list of layers:

- first layer - orders matching logic, that layer validate orders, gather tokens from users and trigger position creation
- second layer - core, that layer contain general postions lifecycle logic (creation, execution, cancellation).

1.5 PROJECT DASHBOARD

Client	Opium Network
Audit name	Protocol v1
Initial version	eb1de1afb41c571ca19307d670f53c2cc2756e98 a6df21c436cfdb975bede13f648fa14b5192e6a1
Final version	394e3257f806b5f5add304195be1ca5d1e9255b7 97a3c1d94746461b98d238339e98d1d1af71c95c
SLOC	1900
Date	2020-12-21 - 2021-01-30
Auditors engaged	2 auditors

FILES LISTING

TokenMinter.sol	TokenMinter.sol
Core.sol	Core.sol
OracleAggregator.sol	OracleAggregator.sol
TokenSpender.sol	TokenSpender.sol
Registry.sol	Registry.sol
SyntheticAggregator.sol	SyntheticAggregator.sol
LibCommission.sol	LibCommission.sol
LibDerivative.sol	LibDerivative.sol
UsingRegistry.sol	UsingRegistry.sol
WhitelistedWithGovernance.sol	WhitelistedWithGovern...
Whitelisted.sol	Whitelisted.sol
LibEIP712.sol	LibEIP712.sol
WhitelistedWithGovernanceAndChangableTimelock.sol	WhitelistedWithGovern...
IDerivativeLogic.sol	IDerivativeLogic.sol
IOracleId.sol	IOracleId.sol
HasCommission.sol	HasCommission.sol
PayoutHelper.sol	PayoutHelper.sol
BalanceHelper.sol	BalanceHelper.sol
ExecutableByThirdParty.sol	ExecutableByThirdPart...
SwaprateMatchBase.sol	SwaprateMatchBase.sol
SwaprateMatch.sol	SwaprateMatch.sol
LibSwaprateOrder.sol	LibSwaprateOrder.sol
MatchSwap.sol	MatchSwap.sol

MatchLogic.sol	MatchLogic.sol
Match.sol	Match.sol
MatchPool.sol	MatchPool.sol
LibOrder.sol	LibOrder.sol
MatchCreate.sol	MatchCreate.sol
RegistryErrors.sol	RegistryErrors.sol
OracleAggregatorErrors.sol	OracleAggregatorError...
SyntheticAggregatorErrors.sol	SyntheticAggregatorEr...
MatchingErrors.sol	MatchingErrors.sol
CoreErrors.sol	CoreErrors.sol
UsingRegistryErrors.sol	UsingRegistryErrors.sol
TokenMinter.sol	TokenMinter.sol
ERC7210Transferable.sol	ERC7210Transferable.sol
ERC7210BackwardCompatible.sol	ERC7210BackwardCompat...
ERC7210Composable.sol	ERC7210Composable.sol
ERC7210Base.sol	ERC7210Base.sol
ERC7210Mintable.sol	ERC7210Mintable.sol
IERC7210Receiver.sol	IERC7210Receiver.sol
IERC7210.sol	IERC7210.sol
ObjectsLib.sol	ObjectsLib.sol
UintArray.sol	UintArray.sol
UintsLib.sol	UintsLib.sol
LibPosition.sol	LibPosition.sol

FINDINGS SUMMARY

Level	Amount
Critical	1
Major	1
Warning	4
Comment	2

CONCLUSION

Smart contracts have been audited and several suspicious places have been spotted. During the audit 1 critical issue was found, one issue was marked as major because it could lead to some undesired behavior, also several warnings and comments were found and discussed with the client. After working on the reported findings all of them were resolved or acknowledged (if the problem was not critical). So, the contracts are assumed as secure to use according to our security criteria.Final commit identifiers with all fixes: `394e3257f806b5f5add304195be1ca5d1e9255b7`, `97a3c1d94746461b98d238339e98d1d1af71c95c`

2.FINDINGS REPORT

2.1 CRITICAL

CRT-1	Infinity minting issue
File	ERC7210Transferable.sol
Severity	Critical
Status	Fixed at 97a3c1d9

DESCRIPTION

This issue is about the `ERC7210Transferable` contract's function `_batchTransferFrom` introducing the way to wreck the user's balances allowing for the position holder to transfer a batch of the positions to itself in here: ERC7210Transferable.sol on lines 122,123,124,125,126,127,128.

This can lead to the balances increase/decrease, unexpected by the applcation's logic,
which can only be fixed with contract redeployment and manual data recovery.
Even being called from `safeBatchTransferFrom` in here: ERC7210Transferable.sol#L30
(the function's name is misleading, by the way) it still can lead to the wreckage.

RECOMMENDATION

It is recommended to introduce additional requirement for the function's arguments `_from` and `_to` not to be equal.

2.2 MAJOR

MJR-1	Invalid owner of token
File	ERC721OBackwardCompatible.sol
Severity	Major
Status	Acknowledged

DESCRIPTION

At the line ERC721OBackwardCompatible.sol#L41 `ownerOf` function returns self contract address probably instead of real token owner, so according that function current contract owns all which exists in that contract.

RECOMMENDATION

Return real token owner instead of `address(this)`

CLIENT'S COMMENTARY

> Since it's not ERC721, but simulation of backward compatibility we can't assign a specific owner to the tokenId. We are aware of it, but this can't be fixed

2.3 WARNING

WRN-1	Incorrect token selection
File	ObjectsLib.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

This issue is about potentially incorrect token identifiers can be computed in here: ObjectsLib.sol#L26 and in here: ObjectsLib.sol#L25.

Big enough `_tokenId` param can lead to the incorrect asset bin number calculation result, which is unexpected by the application logic.

RECOMMENDATION

It is recommended to introduce `SafeMath` usage to avoid retrieving incorrect asset bin number.

CLIENT'S COMMENTARY

> Acknowledged, will be fixed in the next release

WRN-2	Incorrect minting/burning token amount
File	ERC7210Composable.sol
Severity	Warning
Status	Fixed at **97a3c1d9**

DESCRIPTION

This issue is about incorrect token amount can be minted or burned with carefully crafted in here: ERC7210Composable.sol#L138, ERC7210Composable.sol#L135, ERC7210Composable.sol#L114, ERC7210Composable.sol#L97.

RECOMMENDATION

It is recommended to introduce `SafeMath` usage to avoid incorrect tokens amount being minted of burned with `diff.mul(_quantity)`, `diff.mul(_quantity)`, `_finalTokenRatio[index].mul(_quantity)`, `_initialTokenRatio[index].mul(_quantity)` respectively.

WRN-3	Incorrect oracle data fetch response
File	OracleAggregator.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

This issue is about potentially incorrect oracle data fetch response from function `recursivelyFetchData` in here: OracleAggregator.sol#L58. Fetching oracle data with `period` and `times` arguments big enough can lead to the data being fetched for the wrong timestamp set.

RECOMMENDATION

It is recommended to also check if all the timestamps requested are in the ascending order in here: OracleAggregator.sol#L61

CLIENT'S COMMENTARY

> Acknowledged, will be fixed in the next release

WRN-4	Wrong operator position
File	MatchLogic.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

At the line MatchLogic.sol#L195 we have `.add(1)` operation after previous division. It's not clear about purposes of that addition. Makes sense if addition applied for `_denominator`:

```
_numerator.mul(PERCENTAGE_BASE).div(_denominator.add(1))
```

in that case `getDivisionPercentage` will never fail with div by zero error.

RECOMMENDATION

Add more comments to clarify or apply `.add(1)` for `_denominator`

CLIENT'S COMMENTARY

> Acknowledged, not needed there anymore, will be removed in the next release

2.4 COMMENTS

CMT-1	Useless expensive checks
File	ERC7210Composable.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

In function `decompose` defined at ERC7210Composable.sol#L30 there are two checks:

```
require(_tokenIds.length > 0, "TOKEN_MINTER:WRONG_QUANTITY");
require(_tokenIds.isUnique(), "TOKEN_MINTER:TOKEN_IDS_NOT_UNIQUE");
```

at lines ERC7210Composable.sol#L33-L34.
This checks are useless due to at line ERC7210Composable.sol#L41 we have check that guarantees that `_tokenIds` is equal to tokens array that was used in `compose` method, which means that `_tokenIds` already satisfy requirements.

Same commentary is applicable for lines:

- ERC7210Composable.sol#L62
- ERC7210Composable.sol#L60

RECOMMENDATION

We suggest to remove checks to save gas

CLIENT'S COMMENTARY

> Agree, acknowledged, will be fixed in the next release

CMT-2	Use mapping instead of array
File	Whitelisted.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

In contract `Whitelisted` defined at Whitelisted.sol array type used to store whitelist, so if we need check that user is whitelisted we have to spend O(N) operations, so it would be much more effective using mapping type.

RECOMMENDATION

We suggest to use mapping type instead of array

CLIENT'S COMMENTARY

> Acknowledged, will be fixed in the next release

3.ABOUT MIXBYTES

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

BLOCKCHAINS

Ethereum

Cosmos

EOS

Substrate

TECH STACK

Python

Solidity

Rust

C++

CONTACTS

https://github.com/mixbytes/audits_public

https://mixbytes.io/

hello@mixbytes.io

https://t.me/MixBytes

https://twitter.com/mixbytes