BEBOP SECURITY AUDIT REPORT

July 27, 2023

MixBytes()

TABLE OF CONTENTS

1. INTRODUCTION 2

1.1 Disclaimer 2

1.2 Security Assessment Methodology 2

1.3 Project Overview 6

1.4 Project Dashboard 6

1.5 Summary of findings 8

1.6 Conclusion 9

2.FINDINGS REPORT 10

2.1 Critical 10

2.2 High 10

H-1 EIP712 DOMAIN_SEPARATOR stored as immutable 10

2.3 Medium 11

M-1 Unjustified unlimited approvals 11

M-2 Potentially unsafe usage of `ecrecover` return value 12

M-3 Truncation of the `nonce` value 13

M-4 Extra msg.value could be lost 14

2.4 Low 15

L-1 Consequent `WETH` `deposit`/`withdraw` 15

3. ABOUT MIXBYTES 16

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

Bebop is a blockchain platform enhancing the efficiency of token transactions by offering more than standard one-to-one swaps. The platform allows users to sell or buy multiple tokens in a single transaction. With the introduction of Bebop's Aggregation feature, a single taker can match with potentially numerous makers, maximized by the block resource limit. Furthermore, Bebop V2 allows the distribution of individual sections of basket trades among several makers, optimizing the overall price for the taker and ensuring the best value in their transactions.

1.4 Project Dashboard

Project Summary

Title	Description
Client	Bebop
Project name	Bebop
Timeline	10.07.2023 - 27.07.2023
Number of Auditors	2

Project Log

Date	Commit Hash	Note
10.07.2023	f42cc5efe7a843bb015aec409aaa08e7643d0ca9	initial commit for the audit
19.07.2023	c5c9b7fa4c496e768ef1261d0da6e9cd5bd449f8	commit for the reaudit

Project Scope

The audit covered the following files:

File name	Link
src/contracts/BebopSettlement.sol	BebopSettlement.sol
src/contracts/base/BebopSigning.sol	BebopSigning.sol
src/contracts/base/BebopTransfer.sol	BebopTransfer.sol
src/contracts/libs/Commands.sol	Commands.sol
src/contracts/libs/Order.sol	Order.sol
src/contracts/libs/Signature.sol	Signature.sol
src/contracts/libs/Transfer.sol	Transfer.sol
src/contracts/libs/common/BytesLib.sol	BytesLib.sol
src/contracts/libs/common/SafeCast160.sol	SafeCast160.sol

Deployments

Polygon

Contract	Address	tx hash
BebopSettlement	0xbeb09000fa59627dc02bb55448ac1893eaa501a5	0xeceb74224cb29516f98239552360134876531f530afc5c2cb6b7309ce8e1c957

Arbitrum

Contract	Address	tx hash
BebopSettlement	0xbeb09000fa59627dc02bb55448ac1893eaa501a5	0x2015de342fe81705a35950860257bedcc6340bd188190d4063f46d1593d9b7b2

Ethereum Mainnet

Contract	Address	tx hash
BebopSettlement	0xbeb09000fa59627dc02bb55448ac1893eaa501a5	0xa0507db7cb367917ed1b18250081941db242b149aae617090d8483bfa3ac4c78

1.5 Summary of findings

Severity	# of Findings
Critical	0
High	1
Medium	4
Low	1

ID	Name	Severity	Status
H-1	EIP712 DOMAIN_SEPARATOR stored as immutable	High	Fixed
M-1	Unjustified unlimited approvals	Medium	Acknowledged
M-2	Potentially unsafe usage of `ecrecover` return value	Medium	Acknowledged
M-3	Truncation of the `nonce` value	Medium	Fixed
M-4	Extra msg.value could be lost	Medium	Fixed
L-1	Consequent `WETH` `deposit`/`withdraw`	Low	Acknowledged

1.6 Conclusion

During the audit process, 1 high, 4 medium, and 1 low severity issues were discovered. All these findings were fixed or acknowledged by the developers.

2.FINDINGS REPORT

2.1 Critical

Not Found

2.2 High

H-1	EIP712 DOMAIN_SEPARATOR stored as immutable
Severity	High
Status	Fixed in 7fe304e4

Description

To prevent so-called "replay attacks", EIP712 incorporates the chain ID into the signed data. However, the audited BebopSigning.sol#L42 EIP712 implementation uses cached values (stored as immutable variables), independent of the actual chain ID which could be returned by the `CHAINID` EVM opcode. This approach is potentially unsafe as such signatures could be deemed valid in forked networks, thus creating a vulnerability to replay attacks.

When combined with unlimited approvals (refer to Medium.1), orders placed on one network could be "replayed" on a forked network without the account owner's consent.

Recommendation

To prevent replay attacks, it is recommended to adhere to the best practices of the EIP712 implementation (refer to OpenZeppelin's EIP712.sol).

2.3 Medium

M-1	Unjustified unlimited approvals
Severity	Medium
Status	Acknowledged

Description

Several parts of code grant unlimited `type(uint).max` approvals even when it's not justifiable:

- BebopSettlement.sol#L155
- BebopTransfer.sol#L103

No threat is inherent to it. However, it diminishes the overall contract security and could potentially be exploited by various attack vectors.

Recommendation

It's recommended to grant approvals only for the exact amount that is anticipated to be spent.

Client's commentary

Acknowledged. This is to improve the UI/UX so that the user only needs to sign for the first trade of the token, instead of signing each time.

M-2	Potentially unsafe usage of `ecrecover` return value
Severity	Medium
Status	Acknowledged

Description

The `ecrecover` function may return a non-zero value even if the signature is invalid. It's unsafe to use the return value of `ecrecover` for any purpose other than comparing it with a valid signer value. Currently, these parts of the code use it as an index for the `address => boolean map`:

- BebopSigning.sol#L119
- BebopSigning.sol#L137

While it's unlikely that this issue can be exploited in the current code, future modifications could potentially make the issue more severe.

Recommendation

It is recommended to use the `ecrecover` value only for comparison with the expected value, and not as an arbitrary value.

M-3	Truncation of the `nonce` value
Severity	Medium
Status	Fixed in 7fe304e4

Description

In the BebopSigning.sol#L167 function, a `uint256` value is cast to `uint64`, silently ignoring higher bits of the value. This could lead to value collisions when different nonce values as `uint256` have the same values as `uint64`.

Although the `uint64` value range is generally sufficient for the purpose of a `nonce`, and a `nonce` collision would likely cause a revert and, thus, it doesn't seem to be exploitable, the code's security could be further enhanced by addressing this issue.

Recommendation

It's recommended to utilize as many bits of the value as possible (248 bits seems to be a reasonable approach for this code). Additionally, it's recommended to ensure that the truncated bits of the value were actually zero to prevent value collisions.

M-4	Extra msg.value could be lost
Severity	Medium
Status	Fixed in 7fe304e4

Description

This code ensures that the passed msg.value is large enough, but it ignores any extra msg.value that's passed:

- BebopSettlement.sol#L76
- BebopSettlement.sol#L210

Any extra value passed to the transaction will be lost by the user.

Recommendation

It is recommended to refund the extra msg.value or to revert transactions with an unexpectedly large msg.value.

2.4 Low

L-1	Consequent `WETH` `deposit`/`withdraw`
Severity	Low
Status	Acknowledged

Description

If both `nativeTokens.toMaker` and `nativeTokens.toTaker` are non-zero, the audited code will trigger a consequent `deposit` and then a `withdraw` to the `WETH`:

- BebopSettlement.sol#L77-L90
- BebopSettlement.sol#L211-L223

It can be simplified to a single `deposit` or `withdraw` depending on the difference between the `toMaker` and `toTaker` values.

Recommendation

In order to optimize gas consumption, it is recommended to improve the code in accordance with the issue outlined above.

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