METALEX LEXSCROW SECURITY AUDIT REPORT

Sep 10, 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 MetaLeX protocol allows the customization of Gnosis Safe multisigs to be used as a committee for a DAO. The customization allows restricting actions for the specific multisig via a special Guard contract, which acts as a filter for all txs from the multisig. Apart from the Guard contract, MetaLeX offers a list of different implants that can be installed into the Safe contract to extend its functionality. An example of such an extension is an implant that can be used by multisig owners to create grants.
The current audit focused on the LeXscroW part of the MetaLeX protocol. LeXscroW is a next generation optionally-conditional escrow tool optimized for MetaLeX's cybernetic law philosophy. It enables fully immutable and autonomous non-custodial escrow deployments, which may either be configured for purely onchain execution or as a trust-minimized enforcement mechanism for legal agreements.

1.4 Project Dashboard

Project Summary

Title	Description
Client	MetaLeX
Project name	LeXscrow
Timeline	26.06.2024 - 08.08.2024
Number of Auditors	3

Project Log

Date	Commit Hash	Note
26.06.2024	94ca277528bb25b8421dc127941c18915144eb29	Commit for the audit
19.07.2024	c3cbdb6f051dbce1bae237c20b6bc3d06a82af78	Commit for the re-audit
08.08.2024	e1dc7239b1c4785453dbf22ca37781e3625a0011	Commit for the re-audit 2

Project Scope

The audit covered the following files:

File name	Link
src/libs/LexscrowConditionManager.sol	LexscrowConditionManager.sol
src/EthLexscrowFactory.sol	EthLexscrowFactory.sol
src/DoubleTokenLexscrowFactory.sol	DoubleTokenLexscrowFactory.sol

File name	Link
src/TokenLexscrow.sol	TokenLexscrow.sol
src/EthLexscrow.sol	EthLexscrow.sol
src/DoubleTokenLexscrow.sol	DoubleTokenLexscrow.sol
src/TokenLexscrowFactory.sol	TokenLexscrowFactory.sol

Deployments

File name	Contract deployed on mainnet	Comment
TokenLexscrowFactory.sol	0x61aEF2...358009c7	
EthLexscrowFactory.sol	0x4C487a...d5660936	
DoubleTokenLexscrowFactory.sol	0x3dd706...f0625B93	

1.5 Summary of findings

Severity	# of Findings
Critical	2
High	0
Medium	11
Low	16

ID	Name	Severity	Status
C-1	Amount withdrawable by buyer can be set to a smaller value	Critical	Fixed
C-2	A new `buyer` address is not assigned if the previous one was rejected	Critical	Fixed
M-1	Possible DoS with small transfers	Medium	Fixed
M-2	The `buyer` address can be set to the `seller` address	Medium	Fixed
M-3	Random depositor `amountDeposited` is not cleared	Medium	Fixed
M-4	Incorrect fee calculation	Medium	Fixed
M-5	Conditions' order impact	Medium	Acknowledged
M-6	Tests do not work	Medium	Fixed
M-7	Double escrow for one token	Medium	Fixed
M-8	Centralization risks	Medium	Fixed

M-9	Escrows can be reused	Medium	Acknowledged
M-10	`rejectDepositor` can be front-run	Medium	Fixed
M-11	Incorrect usage of `permit`	Medium	Fixed
L-1	Unused errors	Low	Fixed
L-2	Redundant checks for contract code length	Low	Fixed
L-3	Missing check for 0 value transferred	Low	Fixed
L-4	4 Missing checks for zero address	Low	Fixed
L-5	Deposits with 0 amounts	Low	Fixed
L-6	Unnecessary deadline check	Low	Fixed
L-7	Use OZ libs instead of Solady	Low	Acknowledged
L-8	Decimals are checked without staticcall	Low	Fixed
L-9	No two-step address change for `buyer` and `seller`	Low	Acknowledged
L-10	Unnecessary event emitted if escrow has already expired	Low	Fixed
L-11	`getReceipt` can be called by anyone with arbitrary parameters	Low	Acknowledged
L-12	Missing uniqueness and inequality checks	Low	Fixed
L-13	Unnecessary `payable` function	Low	Fixed
L-14	Escrows are incompatible with rebaseable tokens	Low	Fixed
L-15	Unnecessary usage of the `Math` library	Low	Fixed
L-16	Incorrect interface ID	Low	Fixed

1.6 Conclusion

Apart from the list of findings presented in the report during the audit, we checked the next attack vectors:

1. **Possible reentrancy attacks.** The `ReentrancyGuard` library is employed in all critical sections of the code. The `nonReentrant` modifier is in place to ensure that that reentry into key functions such as deposits, withdrawals, and escrow execution is not possible. A review confirmed that when an ERC777 token is used as a deposit token, there is no risk of reentrancy attacks via transfer hooks.

2. **Poisoned tokens.** It has been verified that poisoned tokens cannot be leveraged in attacks against the system. There are no cases where a poisoned token can be used to break the logic of legitimate escrow contracts. Users must agree to buy poisoned tokens with zero liquidity. The system safeguards against any unintended interactions with poisoned tokens.

3. **Escrow offers are correctly implemented.** For open offers, the first user to deposit the required funds is designated as the seller or buyer. In closed offer scenarios, only a specified buyer is permitted to make a deposit. All withdrawals of any excess or rejected tokens are processed accurately. Fees are correctly allocated to the intended recipient upon offer fulfilment. If the offer has expired, all the accounting is handled appropriately.

4. **Deposit frontrunning.** It is possible to front-run any deposit made to the protocol, but it will not cause any damage to the protocol or users. In case when a deposit with permit call is front-run, the funds are still correctly attributed to the current buyer or seller. If a user attempts to deposit the entire `totalWithFee`, they could be front-run by a small deposit which will lead to their transaction being reverted. However, this approach is not economically viable for an attacker, and any funds they deposit are still recorded as a buyer deposit.

5. **Tokens without permit functionality.** A particular scenario was examined where tokens lacking a `permit` function and with an empty fallback do not revert on a call to `depositTokensWithPermit`. In such cases, users' funds remain secure. A deposit transaction will only revert if there has not been adequate approval granted to the escrow contract by the user.

6. **Correct calculations.** All previously unchecked blocks have been thoroughly checked. Critical areas susceptible to overflows or underflows have been scrutinized to prevent such occurrences. The fee calculation process has undergone a thorough review, and suggestions for enhancements have been outlined in the subsequent audit report.

7. **Function access restrictions.** It has been verified all the functions requiring restricted access have all the necessary checks. The system ensures that seller, buyer, or recipient addresses cannot be altered maliciously.

2.FINDINGS REPORT

2.1 Critical

C-1	Amount withdrawable by buyer can be set to a smaller value
Severity	Critical
Status	Fixed in c3cbdb6f

Description

There is an issue at the TokenLexscrow.sol#L515. An attacker can deposit 1 wei of token used in an escrow directly to the contract and call to `checkIfExpired`. If `refundable == true`, `amountWithdrawable[buyer]` will be set to 1 wei (the amount deposited by the attacker). The same issue can be found EthLexscrow.sol#L370. While it's more difficult to exploiton Ethereum, it's easier on other EVM-compatible chains via `selfdestruct` to send 1 wei of native tokens.

Recommendation

We recommend changing `=` to `+=`. It will account for all additional `amountWithdrawable` mapping changes.

Client's commentary

> Fixed as recommended in 6652b1b3

C-2	A new `buyer` address is not assigned if the previous one was rejected
Severity	Critical
Status	Fixed in c3cbdb6f

Description

There is an issue at the EthLexscrow.sol#L326. It resets the `buyer` address, but the `deposited` variable value remains unchanged. It will prevent the new `buyer` address from being set, as the check at the EthLexscrow.sol#L237 will fail. If the escrow is expired, then `amountWithdrawable[address(0)]` will be set to a non-zero value (at EthLexscrow.sol#L370), effectively making the funds get stuck on the contract.

Recommendation

We recommend resetting the `deposited` variable value whenever the `buyer` address is reset.

Client's commentary

> Fixed in 6943bc52: have resetted the `deposited` variable when the balance - `pendingWithdraw` falls below the `deposit` amount as well as when an `openOffer` depositor is rejected; the first condition should cover both, but kept the latter for belt-and-suspenders to ensure `deposited` is properly deleted. The `buyer` address is not reset in a non-open offer anyway unless the LeXscrow executes (in which case `deposited` is already deleted).

2.2 High

Not Found

2.3 Medium

M-1	Possible DoS with small transfers
Severity	Medium
Status	Fixed in c3cbdb6f

Description

A vulnerability exists at the TokenLexscrow.sol#L335, where minor direct deposits to the contract could trigger the check to pass, which will make users' deposits revert.

Recommendation

We recommend transferring only the necessary amount from the depositor's address if attempting to deposit more than `totalWithFee`. It will prevent unexpected reverts and won't require users to calculate the exact amount considering possible dust on the contract.

Client's commentary

> Agreed, this is a UX improvement. Fixed in 75758bc7 by reducing `_amount` by the surplus, or reverting if `_amount` is less than the surplus.

M-2	The `buyer` address can be set to the `seller` address
Severity	Medium
Status	Fixed in c3cbdb6f

Description

There is an issue within the EthLexscrow.sol#L264. Currently, the `buyer` can alter the `buyer` variable value to the `seller`'s address. It will lead to the `seller`'s `amountWithdrawable` value being reduced at the EthLexscrow.sol#L369. The remaining tokens would be stuck on the contract as the `balance` was accounted in the `pendingWithdraw` variable.

Recommendation

We recommend adding checks that `_buyer != seller` and `_buyer != buyer` to the `updateSeller` and `updateBuyer` functions.

Client's commentary

> Fixed in all three LeXscrow patterns (apologies for formatting updates) in 2480ace6.

M-3	Random depositor `amountDeposited` is not cleared
Severity	Medium
Status	Fixed in c3cbdb6f

Description

There is an issue inside the `receive` function defined at the EthLexscrow.sol#L230. When `openOffer` is `false` and `msg.sender` is not the `buyer`, a third party may deposit on behalf of the `buyer`. After an`execute` call, the depositor's`amountDeposited` is not reset to 0. This can lead to a situation where, upon rejection of the depositor, the `pendingWithdraw` exceeds the`address(this).balance`. The subsequent `buyer` deposit could then be less than the actual amount (if the deposit is sufficient not to revert at the EthLexscrow.sol#L231).

Recommendation

We recommend disallowing deposits from the random addresses if the `openOffer` value was set to `false`.

Client's commentary

> Because the ability for random addresses (such as fee-earning contracts) to deposit amounts is important for some usecases, I've taken an alternate solution path in 0c6c68e4 by instead simply always crediting the deposited amount to the `buyer`'s `amountDeposited` for non-open offers, as the buyer is receiving the benefit of the deposited amount anyway. The tradeoff is that `seller` can effectively only reject `buyer` via `rejectDepositor`, which is acceptable because the primary purpose for that function is to handle undesired `openOffer` buyers (as depositors for non-open offers can be reasoned as affiliated with the `buyer` in some way).

M-4	Incorrect fee calculation
Severity	Medium
Status	Fixed in c3cbdb6f

Description

A potential rounding error exists at the TokenLexscrowFactory.sol#L117. The `_feeDenominatorAdjusted` may round down to 0 if a token's `_decimals` are significantly less than 18, leading to the ONE constant being used as `_feeDenominatorAdjusted`. This results in the fee equating to `_totalAmount` at the TokenLexscrowFactory.sol#L123 Conversely, if `_decimals` exceed 18, `_feeDenominatorAdjusted` does not accurately represent a fraction of `_totalAmount`, resulting in a lower-than-expected `_fee`. Additionally, tokens with decimals = 0 will encounter a similar issue, as `_feeDenominatorAdjusted` will not reflect the correct proportion of `_totalAmount`.

Recommendation

We recommend introducing basis points as the denominator for fee calculation and specifying the desired share of `_totalAmount` as the numerator. The fee calculation would then be:

```
_fee = _totalAmount * _fee_share / BASIS_POINTS;
```

It will lead to increased precision during the fee calculation.
We also recommend not using token decimals in the fee calculation.

Client's commentary

> Updated to basis points calculations and removed decimals in 8ae94cf0

M-5	Conditions' order impact
Severity	Medium
Status	Acknowledged

Description

The current implementation of the conditions check relies on the condition order, which makes the system more vulnerable to the owner's faults and makes the protocol less flexible.
LexscrowConditionManager.sol#L48-L60

Recommendation

We recommend redesigning the conditions check architecture to make its order independent.

Client's commentary

> I believe the architecture should already be order independent regardless of the `Logic` of the pertinent `Condition` struct.
> In order to match the syntax and architecture of the BORG-Core Conditions, I have made harmonizing changes to the conditions in 76c2ce71

M-6	Tests do not work
Severity	Medium
Status	Fixed in e1dc7239

Description

Currently, the protocol's tests do not work, making it impossible to check test coverage and prepare PoCs.

Recommendation

We recommend prioritizing, preparing, and setting up all the tests before the protocol deployment.

Client's commentary

> All test are updated and passing for all contracts as of c3cbdb6f.

M-7	Double escrow for one token
Severity	Medium
Status	Fixed in c3cbdb6f

Description

If `_tokenContract1 == _tokenContract2` in the DoubleTokenLexscrow, all deposited funds will be blocked on the contract: DoubleTokenLexscrow.sol#L243-L244.

Recommendation

We recommend adding a check that `_tokenContract1` is not equal to `_tokenContract2`.

Client's commentary

> Fixed in 3022cf9b.

M-8	Centralization risks
Severity	Medium
Status	Fixed in c3cbdb6f

Description

`receiver` can front-run escrow creation and change fees: DoubleTokenLexscrowFactory.sol#L192-L204, EthLexscrowFactory.sol#L124-L131, TokenLexscrowFactory.sol#L147-L154

Recommendation

We recommend using a 2-step fee update with a mandatory time delay to mitigate this risk, ensuring transparency in fee adjustments.

Client's commentary

> Updated to two-step fee updates with a 24 hour delay in 8ae94cf0

M-9	Escrows can be reused
Severity	Medium
Status	Acknowledged

Description

Currently, all escrows can be reused. However, the logic of a condition cannot guarantee that it will be correct for two consecutive executions.

Recommendation

We recommend restricting the usage of the escrows more than one time.

Client's commentary

> The nature of conditions (by design) is that they may not be the same for consecutive executions, for example a price condition. Comments have been updated 76c2ce71 – especially LexscrowConditionManager.sol#L68, to highlight this

M-10	`rejectDepositor` can be front-run
Severity	Medium
Status	Fixed in c3cbdb6f

Description

`rejectDepositor` can be front-run by the `buyer`, so the `seller` tx will revert: TokenLexscrow.sol#L455-L473, EthLexscrow.sol#L311-L329.

Recommendation

We recommend allowing the seller to pause the `updateBuyer` function for such cases.

Client's commentary

> c1043344 together with the 2.3.3 fix handles this; because the `seller` is only able to reject the `buyer`, any front-running will result in the `seller` still rejecting the `buyer`'s deposit. The `buyer` is now prevented from depositing or using this address again so it must replace with a new address. A risk of malicious front-run of spamming new buyer deposits is now documented as a tradeoff of using open offers – the spamming depositor risks their own funds by doing this anyway, as a spammed seller can simply abandon this LeXscrow if they no longer want to accept this tradeoff.

M-11	Incorrect usage of `permit`
Severity	Medium
Status	Fixed in e1dc7239

Description

In the `depositTokensWithPermit` function, a user can sign a permit for a specific `_amount` value, which TokenLexscrow.sol#L320 inside the function which will lead to a revert of the `permit` call: TokenLexscrow.sol#L342

Recommendation

We recommend using the initial `_amount` value for the permit call.

Client's commentary

> Fixed in 27594b58. Also in abadee9e made the same `_surplus` concept changes for excess deposits in DoubleTokenLexscrow as were made in TokenLexscrow (see 2.3.1 Finding and related Fix in TokenLexscrow) for conformity and UX/to prevent DoS similarly (and updated tests).

2.4 Low

L-1	Unused errors
Severity	Low
Status	Fixed in c3cbdb6f

Description

There are unused errors at the following lines:
DoubleTokenLexscrow.sol#L188
EthLexscrow.sol#L170
TokenLexscrow.sol#L241.

Recommendation

We recommend removing the unused errors.

Client's commentary

> Fixed in 30075626

L-2	Redundant checks for contract code length
Severity	Low
Status	Fixed in c3cbdb6f

Description

There is a check for contracts following the ERC20 interface: DoubleTokenLexscrow.sol#L267.

It has redundant checks for contract code lengths: if there is `true` and non-zero length data returned by a `staticcall`, the contract code length is not zero.

Recommendation

We recommend removing the unnecessary code length checks and using general calls to the token contracts by the ERC20 interface, which guarantees reversion if the contract does not implement such an interface.

Client's commentary

> Fixed in 30075626

L-3	Missing check for 0 value transferred
Severity	Low
Status	Fixed in c3cbdb6f

Description

There is a missing check for `fee != 0`:
EthLexscrow.sol#L292.

Recommendation

We recommend adding a check that the transferred `fee` value is not zero.

Client's commentary

> Fixed in 30075626

L-4	4 Missing checks for zero address
Severity	Low
Status	Fixed in c3cbdb6f

Description

If `openOffer = true`, the buyer's address should be zero:
EthLexscrowFactory.sol#L95.

Recommendation

We recommend adding a check that the `buyer` address is zero if `openOffer = true`.

Client's commentary

> Fixed in 30075626 by only assigning the address if `openOffer` == false rather than requiring the zero address input, for simplicity as the unassigned address will be address(0). Also true of the seller address in DoubleTokenLexscrow.

L-5	Deposits with 0 amounts
Severity	Low
Status	Fixed in c3cbdb6f

Description

It is possible to call deposit functions with 0 amounts:
TokenLexscrow.sol#L324
TokenLexscrow.sol#L366.

Recommendation

We recommend adding checks that the deposited amount is not zero.

Client's commentary

> Fixed in e9b9928c

L-6	Unnecessary deadline check
Severity	Low
Status	Fixed in c3cbdb6f

Description

There is a `deadline` check at line TokenLexscrow.sol#L338. A `deadline` check is performed inside the `permit` function, so there is no need to do it outside of that function.

Recommendation

We recommend removing unnecessary `deadline` checks.

Client's commentary

> Fixed in 30075626

L-7	Use OZ libs instead of Solady
Severity	Low
Status	Acknowledged

Description

It is better to use the OZ library instead of the Solady library because it is used more widely and has a lot of battle-tested code:

TokenLexscrow.sol#L56-L142.

Recommendation

We recommend using the OpenZeppelin library instead of the Solady library, which is more widely adopted and has extensive community support.

Client's commentary

> New commits per this report have used OpenZeppelin libraries (for example the updated fee calculations in the 2.3.4 fix), but we are comfortable with the Solady library items used here (mostly transfers) due to their simplicity and extensive overlap with the OpenZeppelin implementations.

L-8	Decimals are checked without staticcall
Severity	Low
Status	Fixed in c3cbdb6f

Description

There is a call to the `decimals` function at the TokenLexscrowFactory.sol#L110.

Recommendation

We recommend performing that call via `staticcall`, which allows to check if the call succeeded.

Client's commentary

> Decimals removed entirely in 8ae94cf0

L-9	No two-step address change for `buyer` and `seller`
Severity	Low
Status	Acknowledged

Description

There are two functions EthLexscrow.sol#L255 and EthLexscrow.sol#L264. That functions allow `seller` and `buyer` to change their addresses in one step. It may lead to setting their addresses to unwanted values.

Recommendation

We recommend introducing a two-step `seller` and `buyer` address change where the new `seller` and `buyer` must accept the changes.

Client's commentary

> The lack of a two-step change is intentional, as it enables either party to designate a custodied address (such as an auto-offramping address) as recipient because such a replacement address does not need to call a function to confirm its designation. All contracts have ability to mutually terminate in the event of mistaken address substitution if the parties' related agreement so provides.

L-10	Unnecessary event emitted if escrow has already expired
Severity	Low
Status	Fixed in c3cbdb6f

Description

There is an issue at the EthLexscrow.sol#L350. This check allows entry into the `checkIfExpired` function, which emits the `TokenLexscrow_Expired` event (or `EthLexscrow_Expired` in case of `EthLexscrow`), even if that function has already been called and `isExpired` was set to `true`.

Recommendation

We recommend changing the following TokenLexscrow.sol#L494 to `if (expirationTime <= block.timestamp && !isExpired)`.

Client's commentary

> addressed in 322ede6e for DoubleTokenLexscrow, and 49d7cc88 for EthLexscrow and TokenLexscrow

L-11	`getReceipt` can be called by anyone with arbitrary parameters
Severity	Low
Status	Acknowledged

Description

There is a `getReceipt` function defined at the TokenLexscrow.sol#L445. This function can be called by anyone and an arbitrary `_tokenAmount` can be passed as a parameter.

Recommendation

We recommend restricting the `getReceipt` function to being called by some specific addresses or revising the method by which `_tokenAmount` is passed.

Client's commentary

> This is intentional as `getReceipt` is purely informational–because it has no bearing on the execution of any LeXscrow type and simply returns a calculation, restrictions were seen as unnecessary code bloat.

L-12	Missing uniqueness and inequality checks
Severity	Low
Status	Fixed in c3cbdb6f

Description

There is a missing check if the conditions are unique:
LexscrowConditionManager.sol#L33.

There is a missing check that `buyer != seller`:
EthLexscrowFactory.sol#L90.

Recommendation

We recommend adding checks for conditions and `buyer` and `seller` addresses.

Client's commentary

> Fixed as to the addresses in e9b9928c, and fixed as to the conditions in 76c2ce71.

L-13	Unnecessary `payable` function
Severity	Low
Status	Fixed in c3cbdb6f

Description

The `DoubleTokenLexscrow` constructor doesn't use `msg.value` so it is unnecessary to mark the function as `payable`: DoubleTokenLexscrow.sol#L238-L248.

Recommendation

We recommend removing the `payable` keyword.

Client's commentary

> Fixed in 30075626 and also for TokenLexscrow.

L-14	Escrows are incompatible with rebaseable tokens
Severity	Low
Status	Fixed in c3cbdb6f

Description

Currently, token escrows are incompatible with rebasable tokens because all positive rebase will be stuck on the contract until the buyer or seller calls the `checkIfExpired` function.

Recommendation

We recommend clearly stating in the documentation that rebasable tokens cannot be used in escrows.

Client's commentary

> Already present in the README, but also fixed for DoubleTokenLexscrow 322ede6e, along with a simple mutual early termination mechanism to save improper ERC20s in limited circumstances as well as general deployment errors. Comment updated in TokenLexscrow 30075626

L-15	Unnecessary usage of the `Math` library
Severity	Low
Status	Fixed in e1dc7239

Description

The `_mulDiv` function from the `Math` library is used in all factories, but it only overcomplicates fee calculation which is tolerant to calculation imprecision: EthLexscrowFactory.sol#L167-L246

Recommendation

We recommend not using the `_mulDiv` method but rather using general solidity operations.

Client's commentary

> Fixed in e1dc7239

L-16	Incorrect interface ID
Severity	Low
Status	Fixed in e1dc7239

Description

It's unclear what the interface with such an ID (`_INTERFACE_ID_BASE_CONDITION = 0x8b94fce4`) is because the `ICondition` interface has a different interface ID: LexscrowConditionManager.sol#L57

Recommendation

We recommend using `type(ICondition).interfaceId` instead of `0x8b94fce4`.

Client's commentary

> Fixed in e1dc7239

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