HashEx

x0Swap

smart contracts audit report

Prepared for:

https://x0swap.com

Authors: HashEx audit team

August 2021

Contents

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and HashEx and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (HashEx) owe no duty of care towards you or any other person, nor does HashEx make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties, or other terms of any kind except as set out in this disclaimer, and HashEx hereby excludes all representations, warranties, conditions, and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, HashEx hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against HashEx, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of the use of this report, and any reliance on this report.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

Introduction

HashEx was commissioned by the `x0swap` to perform an audit of their smart contracts. The audit was conducted between August 28 and September 08, 2021.

The code located in @x0swap/smartcontract GitHub repository was audited after 29a6ece commit.
Update: x0swap team has responded to this report. Updated code is located after ef60315 commit.

The purpose of this audit was to achieve the following:

- Identify potential security issues with smart contracts.
- Formally check the logic behind given smart contracts.

Information in this report should be used for understanding the risk exposure of smart contracts, and as a guide to improving the security posture of smart contracts by remediating the issues that were identified.

Contracts overview

`x0Token`

ERC20-Token with limited supply.

`x0Maker`

Converts Pair tokens to `x0Token` sending them to `x0Bar` address.

`x0Bar`

ERC20-Compatible proxy of burnable contract, which is made to make some additional income holding `x0` tokens on the contract. Fork of SushiBar.

`Timelock`

Implements delayed transaction execution.

`XusdRebaser2`

Contract, which is only allowed to call bar.rebase to change bar's parameters according to the current price.

`UniswapV2Factory`

Implementation of the UniSwap's Factory contract.

`UniswapV2Router02`

Implementation of the UniSwap's Router V2 contract with an extra function.

`Farming`

The MasterChef contract gives out a constant number of X0 tokens per block. It is the only address with minting rights for X0.

`XusdDelegate3`

Implementation of the x0Bar contract.

Found issues

ID	Title	Severity	Status
01	XusdToken: Admin’s extra powers	High	Fixed
02	x0Token: Transfer delegates	High	Fixed
03	XusdDelegate3: Admin’s extra powers	High	P/Fixed
04	DevFundLock: Admin’s extra powers	High	Fixed
05	X0Rebaser2: Not corresponding restrictions	High	Fixed
06	Farming: Admin’s extra powers	High	Fixed
07	Farming: Admin’s extra powers	High	Fixed
08	XusdRebaser2: Input data is not checked	Medium	Fixed
09	UniswapV2Router02: The Front-Running Attack	Medium	Fixed
10	X0Maker: Input data is not checked	Medium	Fixed
11	X0Maker: Malicious Factory	Medium	Fixed
12	DevFundLock: Input data is not checked	Medium	Fixed
13	XusdRebaser2: Input data is not validated	Medium	Fixed
14	XusdDelegate3: Missing an event	Medium	Fixed
15	Farming: Supporting deflationary tokens	Medium	Informed
16	Farming: Reset user’s rewards	Medium	Fixed
17	Farming: Unclear calculation	Medium	Informed
18	Farming: LP creation	Medium	Fixed
19	Farming: Input data is not checked	Medium	Fixed
20	Farming: Input data is not checked	Medium	Fixed
21	UniswapV2Router02: Useless checking	Low	Fixed

22	UniswapV2Router02: Redundant calls	Low	Fixed
23	XusdRebaser2: Error message	Low	Fixed
24	XusdRebaser2: Code Style	Low	Fixed
25	XusdRebaser2: Running out of gas	Low	P/Fixed
26	XusdRebaser2: Useless logging	Low	Fixed
27	X0Maker: Redundant dependencies	Low	Fixed
28	X0Maker: Immutable parameter	Low	Fixed
29	x0Token: extra definition	Low	Fixed
30	XusdRebaser2: Using experimental features	Low	Fixed
31	XusdDelegate3: Commented code is not deleted	Low	Fixed
32	XusdRebaser2: Two similar functions	Low	Fixed
33	Farming: Input data is not checked	Low	Fixed
34	Farming: Input data is not checked	Low	Fixed
35	Farming: Common recommendations	Low	Fixed

#01 XusdToken: Admin's extra powers High

delegate.sol#L1181: function `rescueTokens` allows the admin to withdraw any amount of any token including the `x0token`. It might be considered as an exit-scum vulnerability since the governance address is able to call the function without any restriction.

Recommendation: We recommend not to make it possible to withdraw the `x0token` via a rescuing mechanism. It's also recommended to make the withdrawing amount limited.

Update: The issue was fixed by removing the function.

#02 x0Token: Transfer delegates High

`_moveDelegates()` function in L972 of `x0Token` contract is designed to be used with every token transfer. However, in `x0Token` it's called with minting new tokens, but not with the usual transfers. The origin (SushiToken) has a warning (L8) about this issue. Issue allows to mint any

number of delegation votes.

Recommendation: stick with the original governance logic and move delegates with transfers.

Update: The issue was fixed. In updated code functions transfer and trasferFrom were overridden where _moveDelegates() is used.

#03 XusdDelegate3: Admin's extra powers — High

Function mint is restricted by a modifier `onlyMinter`, which allows minting by five addresses. At the time of audit some of them are EOA addresses.

Recommendation: We recommend not to allow an EOA address to be able to mint tokens. Even if it is the Governance account.

Update: In the updated code the modifier `onlyMinter` allows to mint only to rebaser, farming and governance addresses.

#04 DevFundLock: Admin's extra powers — High

Dev address is able to change `WITHDRAW_INTERVAL` and `WITHDRAW_HALF_THRESHOLD`. The owner of the contact can update the parameters so locked tokens can be withdrawn to the dev address immediately. Owner also has the possibility to change the dev address.

Recommendation: We recommend setting restrictions values `WITHDRAW_INTERVAL` and `WITHDRAW_HALF_THRESHOLD` since that affects the withdrawal.

Update: The issue was fixed. Values WITHDRAW_INTERVAL and WITHDRAW_HALF_THRESHOLD are declared as immutable in the updated code. Setters, which changed the values, are removed as well.

#05 X0Rebaser2: Not corresponding restrictions — High

rebase.sol#L1153: `require` checks whether a new `xusdScalingFactor` is greater than the `maxScalingFactor`, whereas inside the `bar.rebase(...)` such case is already handled. Consequence of this issue is inability to call rebase to set `xusdScalingFactor` as `maxScalingFactor` when it makes `xusdScalingFactor` greater than the `maxScalingFactor.`

Recommendation: Remove the restriction from `XusdRebaser2` contract.

Update: The issue is fixed since in the updated code the checking is removed.

#06 Farming: Admin's extra powers — High

At line 468 the `owner` can set a value of `x0PerSecond` which defines rate of emission, which is not restricted. It makes it possible for the `owner` to change the rate to any value. Setting the emission rate to a big number can devalue the token.

Recommendation: We recommend capping such parameters of contracts since they affect the emission.

Update: Fixed. New value is checked.

#07 Farming: Admin's extra powers — High

`Owner` is able to `withdraw` all `x0` tokens from the contract by calling the function `emergencyX0()` which is made to save x0 transferring that to a "new Factory". Instead the factory address might be set to any one and then all x0 will be sent to it.

Recommendation: We recommend not to leave a possibility for the admin to withdraw all tokens.

Update: Fixed. Functions `emergencyX0()` and emergencyWithdrawAdmin() are removed.

#08 XusdRebaser2: Input data is not checked — Medium

Input data such as `priceFeed_` (L1052) or `pendingGov_`(L1078) should be checked before assignating.

Recommendation: Add the `require()` checking appropriate restrictions.

Update: New values are checked whether they aren't zero.

#09 UniswapV2Router02: The Front-Running Attack — Medium

Function `addLiquiditySingleToken` does not use slippage parameters such as `amountAMin` and/or `amountBMin`, which are actually a protection against the Front-Running attack. Since in the function there is a call:

```
(amountA, amountB) = _addLiquidity(_path[0], _path[1], leftAmount, rightAmount, 1, 1);
```

with slippage parameters `1, 1;` an attacker is able to send a transaction which would change the price of tokens relative to each other.

Recommendation: Add the slippage parameters to avoid the attack possibility.

Update: Fixed. The slippage parameters are passed in the updated code.

#10 X0Maker: Input data is not checked — Medium

Input data such as `_factory` (L241), `_x0`(L245), `_bar`(L249) should be checked before assignating.

Recommendation: Add the `require()` checking appropriate restrictions.

Update: The issue is fixed in the updated code since function setFactory is removed and other ones check whether set address is not zero.

#11 X0Maker: Malicious Factory — Medium

Owner can change the Factory contract to a malicious contract and steal tokens that are sent with `convertMultiple()` function.

Recommendation: we recommend not allowing changing the factory contract.

Update: Fixed. The function, which allows changing the Factory, is removed.

#12 DevFundLock: Input data is not checked — Medium

In setter functions `setDevAddr()`, `setX0()`, `setWithdrawInterval()` and `setWithdrawHalfThreshold()` input parameters must be checked. In some cases it's needed to be validated (e.g. by calling some view-functions if the parameter is a contract address).

Recommendation: Add the `require()` checking appropriate restrictions and validate the parameters.

Update: Fixed. The setter functions `setDevAddr()`, `setX0()` check whether a new address is not zero. Functions `setWithdrawInterval()` and `setWithdrawHalfThreshold()` are removed.

#13 XusdRebaser2: Input data is not validated — Medium

Addresses, which are passed to `addSyncPairs` aren't checked whether they are actually addresses of pairs. In case if a wrong address will be passed, the function rebase will be broken.

Recommendation: Add a validation of the input addresses. We recommend calling some contact's view-functions to make sure that the addresses are actually the pairs.

Update: Fixed. Pair addresses are checked by calling function `token0()` and `token1()`.

#14 XusdDelegate3: Missing an event — Medium

In L1142 `totalSupply` is changed, but no one event is emitted.

Recommendation: We recommend to emit events `Mint` and `Burn` when the `totalSupply` is changing.

Update: Fixed. Emitting events is implemented.

#15 Farming: Supporting deflationary tokens Medium

Liquidity Pools do not support deflationary tokens. We strongly recommend considering that such tokens will be used within the pools.

#16 Farming: Reset user's rewards Medium

User's rewards will be reset if the owner calls `emergencyWithdrawAdmin` for the user.
Recommendation: remove the `emergencyWithdrawAdmin` function.

Update: Fixed. The function is removed in the new version.

#17 Farming: Unclear calculation Medium

It's unclear from the code and without documentation how the minting amount of xusd tokens should be calculated at L660, 663, also L622 and L625.

Recommendation: add a comment which explains how the minting amount must be calculated.

#18 Farming: LP creation Medium

Adding two pools with the same Liquidity Pool tokens will break reward calculations.

Recommendation: add checking that adding LP token hasn't been added yet.

Update: The issue is fixed by checking if an adding LP-token was already added.

#19 Farming: Input data is not checked Medium

Parameters of the function `config(...)` are not checked. It can be a reason for a braking since the sum of periodShares must be equal `100`.

Recommendation: Check and validate the parameter.

Update: Fixed. The checking is added.

#20 Farming: Input data is not checked Medium

Function `setDevFundDivRate` does not restrict input value from the top. Since the value presents percentage, it must be less or equal to `100`

Recommendation: Check and validate the parameter.

Update: Fixed by adding a require if the input value is less or equal to zero.

#21 UniswapV2Router02: Useless checking Low

L498-499: Checking whether `path[0] == WETH` is useless because no one branch of the if statement uses the fact. Moreover, both of them are making almost the same call: the first one makes the usual `transfer` and the second one - `safeTransfer`.

Recommendation: Remove the if statement and leave the `safeTransfer()` call because the result of it is checked.

Update: Fixed the mentioned issue, but there is almost the same one at L512-513.

#22 UniswapV2Router02: Redundant calls Low

In function `addLiquiditySingleToken` `UniswapV2Library.pairFor()` is called a few times, which is a gas wasting.

Recommendation: Save the first result of the function call in a local variable and use it then to save the gas.

Update: Fixed. The updated code saves the first result of the function call in a local variable and uses it below.

#23 XusdRebaser2: Error message Low

rebase.sol#L1065: An error message should be set clarifying the reason for the error.

Update: Fixed.

#24 XusdRebaser2: Code Style Low

rebase.sol#L1074, L1086, L1099, L1115: some external function names start from '_' or are defined with snake case instead of camel case, which is contrary to the solidity code style.

Recommendation: Follow the Code Style.

Update: The issue was fixed by renaming the functions.

#25 XusdRebaser2: Running out of gas Low

Functions `removeUniPair()`, `removeBalPair()`, `addSyncPairs()` have for-loops through arrays, which do not have restrictions on their size. It makes it possible to run the functions out of gas.

Update: The issue was partially fixed by adding a limit restricting the amount of the pairs. It's only partially fixed because it can still run out of blogGasLimit iterating transactions.

#26 XusdRebaser2: Useless logging Low

L1136: In function `rebase` emits event `MintAmount(mintAmount)`, but value of `mintAmount` is always `0`.

Update: Fixed. The emitting is removed as well as the `mintAmount` variable.

#27 X0Maker: Redundant dependencies Low

Libraries `IUniswapV2ERC20` and `SafeMath128` were declared, but never used.

Recommendation: Clean up libraries and contracts which the contract doesn't depend on.

Update: Fixed.

#28 X0Maker: Immutable parameter Low

Variable `weth` might be immutable to save the gas.

Update: Fixed. The variable is declared as immutable.

#29 x0Token: extra definition Low

If `_maxSupply` was defined as `public`, it wouldn't be necessary to define the function `getCurrentMaxSupply` since a view-function would be implicitly created for `_maxSupply`.

Update: Fixed. The variable is declared as public. The function is removed.

#30 XusdRebaser2: Using experimental features Low

L2: experimental features are used, which is discouraged.

Update: Fixed.

#31 XusdDelegate3: Commented code is not deleted Low

At L800, L824, L856 there are commented lines of code. We recommend deleting source code, which is not used anymore.

Update: Fixed.

#32 XusdRebaser2: Two similar functions Low

Functions `getPrice`, `getCurrentPrice` are almost the same. The only difference are access modifier and two lines which are changing the state(L1281-1283):

```
priceCumulativeLastXusdETH = priceCumulative;
```

```
blockTimestampLast = blockTimestamp;
```

Recommendation: In that case a refactoring is desirable. We recommend to define a `private`(or `internal` function returning thee values: `priceCumulative`, `blockTimestamp` and result of `XusdETHprice.mul(ETHprice)`. The getPrice will call the state and change the state. While the `getCurrentPrice` will just call the new function and return the result of `XusdETHprice.mul(ETHprice)`.

Update: Fixed. The functions are refactored in the updated code.

#33 XusdDelegate3: Typo in comment Low

At line 1132 there is a comment "`positive reabse`" instead of "`positive rebase`".

Update: Fixed.

#34 Farming: Input data is not checked Low

Function `setXusd` lacks checking whether the input parameter is not `0`.

Update: Fixed.

#35 Farming: Common recommendations Low

Constructor of the `Farming` contract lacks checkings of input parameters.

At L285 experimental `pragma` is used, which is not recommended.

`SafeMath` is not used at L594. We recommend using `SafeMath` for each arithmetic operation.

Update: Fixed.

Conclusion

7 high severity and 14 medium severity issues were found.

The audited repository contains no tests. We strongly recommend adding tests to ensure that the contracts work as intended.

Audit includes recommendations on the code improving and preventing potential attacks.

Update: Issues were addressed by the team. Most of the issues were fixed.

Contracts are deployed to the Binance Smart Chain mainnet:

Factory 0x62825171DAb1275cE8D79728d6F1e508d12185DC

Router 0x72326299771BCCc6854f97CcAd54f96691dA1F66

Farming 0xbaCE10Fa0da3C531aF87A384e03D7Df34a724619

DevFundLock 0x92f7128E13e7c8631B03800Ed3d465A3Da081bb6

Timelock 0x86e7071F42b3A11aA09738F2dDCa9BFc8787FF44

GnosisSafeProxy 0x935E12641A3BBdCFC50F4D1C531ed20dF83736a7

X0Maker 0xD69485FF16D6fef0475544f092D5aaA0605927D9

X0Bar 0xB96D05a01dE9A28e6E53170B61C36F1f374DA255 (deployed via proxy, implementation may be changed)

Delegate 0xAc0f79F5538759c94D67b5A6aFa15B6Acef4d6A6

Rebaser 0x69953736AeD6891e55Bd14Bf71E19b56B3A9AD82

X0Token(bridged) 0x63A82Eee817d6FfC0900EAA9E2cB4dcdB827635C (deployed via proxy, current implementation is set to 0xa126c73d1bdf3a3d5f719a8d38a4692186e7503f, which differs from the audited version)

Appendix A. Issues' severity classification

We consider an issue critical, if it may cause unlimited losses or breaks the workflow of the contract and could be easily triggered.

High severity issues may lead to limited losses or break interaction with users or other contracts under very specific conditions.

Medium severity issues do not cause the full loss of functionality but break the contract logic.

Low severity issues are typically nonoptimal code, unused variables, errors in messages. Usually, these issues do not need immediate reactions.

Appendix B. List of examined issue types

Business logic overview

Functionality checks

Following best practices

Access control and authorization

Reentrancy attacks

Front-run attacks

DoS with (unexpected) revert

DoS with block gas limit

Transaction-ordering dependence

ERC/BEP and other standards violation

Unchecked math

Implicit visibility levels

Excessive gas usage

Timestamp dependence

Forcibly sending ether to a contract

Weak sources of randomness

Shadowing state variables

Usage of a deprecated code