HashEx

PreDex
Smart Contracts Security Audit

Prepared for: Predix Network Team

October 19, 2020

By:

HashEx
https://hashex.org

Contents

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as 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 below 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, then 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 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.

Background

HashEx was commissioned by the PreDex team to perform an audit of PreDex smart contracts. The audit was conducted between October 12 and October 19, 2020.

The audited code is located in Hilobrain/PreDex repository. The version used for this audit is a7b0901.

The purpose of this audit was to achieve the following:

- Ensure that smart contracts function as intended.
- Identify potential security issues with smart contracts.

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

Update: the Predix Network Team implemented some of our recommendations and added new features in the code update. We performed the audit recheck. The commit for the audit recheck is 41d74e4.

Critical severity issues

No critical severity issues were found in the smart contracts.

High severity issues

PreDex_POL.sol

Possible double spend attack

`function unstake_lp` is susceptible to reentrancy attack as it does not follow checks-effect-interactions pattern. If by any case a malicious contract is added to the pool it will be able to steal PRDX tokens from the pool. It should be noted that this attack is possible if only a malicious LP token is added to the pool. If the owner of the contract is careful with adding only trusted LP tokens this attack will not be possible. Line 120.

Update: fixed in 41d74e43e198df8cb2dc5590da4df7198d498271.

Medium severity issues

PreDex_Core.sol

1. `function add_prediction_market` can be called twice with the same market id. In such a case all market parameters will be overridden and this will make all predictions made by users on this market irrelevant. It should be noted that this function is callable only by the owner of PreDEX_Core smart contract. Recommendation is to track added markets via introducing a new `mapping` variable to check if a market was already added. Line 147.

   Update: partially fixed in fe45a301945699ad04ceff575b6d539d28e0ef72 by adding a require check for TWAP length is zero. TWAP length must be zero in any pool but the contract itself does not enforce it, so there is a possibility for adding a pool with TWAP length equal zero by mistake.

2. In `function edit_prediction_market` params `uint256 _id`, `address factory`, `address tokenA`, `address tokenB` can be changed. Changing these params can cause all predictions made by users on this market irrelevant. We recommend removing the possibility of editing this market parameters. If any of these parameters needs to be edited, a new market should be added. Line 190.

   Update: fixed in fe45a301945699ad04ceff575b6d539d28e0ef72.

PreDex_Router.sol

1. PreDex_Router contract holds tokens that users use for predictions. It should be noted that the owner of the contract can withdraw all these tokens to an arbitrary address with `function save_erc20`. Users that are using this contract should trust the owner of the PreDex_Router contract. And it is crucially important to secure the owner's account.
2. Variables `predex_contract`, `token_contract`, `POOL_ADDR` should be initialized in the constructor as they are for the contract to work correctly. Lines 53, 57, 61.

PreDex_POL.sol

1. PreDex_POL contract holds tokens that users stake for it. The owner of the contract can withdraw all these tokens to an arbitrary address with `function save_erc20`. Users that are using this contract should trust the owner of the PreDex_POL contract. And it is crucially important to secure the owner's account.
2. The `amount` parameter in `event unstaked` is always zero as it is set to 0 in line 122. Line 126.

   Update: fixed in 41d74e43e198df8cb2dc5590da4df7198d498271.

3. If by any case there are no PRDX tokens in the PreDex_POL contract a user will be unable to unbond staked tokens as `function unstake` will fail on trying to send reward in PRDX tokens to the user. Line 119.

   Update: in the audited version of the code this cannot happen as reward is calculated as part of token balance. So if contract balance is zero, reward will be also zero and token transfer function will not throw an exception. But in the recent code version this problem becomes actual.
4. Pool parameters can be accidentally overridden if a new pool is added with the id that already is in pools array. Recommendation: introduce a new mapping variable to track pool ids that has been added. Line 56.

   Update: fixed in [41d74e43e198df8cb2dc5590da4df7198d498271](41d74e43e198df8cb2dc5590da4df7198d498271).
5. Pair should not be changed in the `function edit_pool` as a different pair actually means a different pool. Line 69.

   Update: fixed in [41d74e43e198df8cb2dc5590da4df7198d498271](41d74e43e198df8cb2dc5590da4df7198d498271).
6. **New**. `function add_reward` can be called by the owner of the contract. If the owner adds a reward with it and does not send an appropriate amount of PRDX tokens to the contract balance, it will cause discrepancies in contract behaviour (see 3.). Line 127.

Low severity issues and informational recommendations

1. The compiler version should be fixed to avoid any potential discrepancies in smart contract behavior caused by different compiler versions. Contracts Babylonian.sol (line 3), Context.sol (line 2), FixedPoint.sol (line 3), IERC20 (line 2), IUniswapV2pair.sol (line 1), Ownable.sol (line 3), SafeMath.sol (line 2), UniswapV2Library.sol (line 1), UniswapV2OracleLibrary.sol (line 1).
2. License is not specified in some of the contracts: UniswapV2Library.sol, UniswapV2OracleLibrary.sol, IUniswapV2pair.sol.
3. Contract names do not follow Solidity naming convention for contracts (contracts and libraries should be named using the CapWords style). Contracts PreDex_Core.sol, PreDex_POL.sol, PreDex_Rewards.sol, PreDex_Router.sol.
4. Function names contracts do not follow Solidity naming convention for functions (functions other than constructors should use mixedCase). Contracts PreDex_Core.sol, PreDex_POL.sol, PreDex_Rewards.sol, PreDex_Router.sol.
5. Variable names do not follow Solidity naming convention for local and state variable names. Contracts PreDex_Core.sol, PreDex_POL.sol, PreDex_Rewards.sol, PreDex_Router.sol.

PreDEX_Core.sol

1. `function save_tokens` is redundant as it's functionality is duplicated in `function save_erc20`. Line 501.

Update: fixed in fe45a301945699ad04ceff575b6d539d28e0ef72.

2. Variables `uint WEEK` and `uint MONTH` are not used anywhere. They can be removed to save gas on contract creation. LInes 88-89.

   Update: fixed in fe45a301945699ad04ceff575b6d539d28e0ef72.
3. Constants `uint HOUR` and `uint YEAR` should have `constant` modifiers. Lines 87, 90.

   Update: fixed in fe45a301945699ad04ceff575b6d539d28e0ef72.
4. Variables `uint256 FACTOR`, `uint256 DECAY_START`, `uint256 MAX_LEV`, `uint256 MIN_AGE` should use mixedCase naming style as UPPER_CASE_WITH_UNDERSCORES style is used for constants, but these variables can be changed by owner of the contract. Lines 92-95.
5. We recommend making `address tokenA` and `address tokenB` parameters in `event NewPair` indexed to make them searchable. Lines 98-99.

   Update: fixed in fe45a301945699ad04ceff575b6d539d28e0ef72.
6. We recommend making `address factory`, `address tokenA` and `address tokenB` parameters in `event PairEdited` indexed to make them searchable. Lines 108-111.

   Update: fixed in fe45a301945699ad04ceff575b6d539d28e0ef72.
7. Gas usage can be reduced in `function update`. Computation `price0Average.mul(1e18).decode144()` is done twice. We recommend introducing a new local variable to store the computation result. Line 351, 353.

   Update: fixed in fe45a301945699ad04ceff575b6d539d28e0ef72.
8. function `get_market_params` is redundant as `function get_market` returns the same parameters besides others. Line 420.
9. function `get_market_fee_perc` is redundant as `function get_market` returns the FEE_PERC parameter besides others. Line 444.
10. Variable `address predex_rewards_addr` is redundant. Line 78.
11. Contract is not fully functional without setting `rewards_contract` variable as it won't be able to calculate rewards in `function close_prediction`. We recommend initializing this variable in the constructor to make the contract fully functional right after deployment. Line 79.
12. We recommend adding an explicit require check `closeTime > block.timestamp` to improve user experience. Line 248.
13. Safe math is not used for uint calculations. Line 311.

    Update: fixed in fe45a301945699ad04ceff575b6d539d28e0ef72.
14. Variable `uint256 prediction_amount` should be renamed to `predictionsNum` as amount suffix may confuse users. Line 42.
15. `function add_prediction_market` is missing validation checks on input parameters. Line 147.

    Update: fixed in fe45a301945699ad04ceff575b6d539d28e0ef72.

16. `function edit_prediction_market` is missing validation checks on input parameters. Line 190.

    Update: fixed in fe45a301945699ad04ceff575b6d539d28e0ef72.
17. `function open_prediction` is missing validation checks on input parameters. Line 238.

    Update: fixed in fe45a301945699ad04ceff575b6d539d28e0ef72.
18. Variables `uint8 REFUND_PERC`, `uint8 POOL_PERC`, `uint8 BURN_PERC` in `struct Market` are using multiplier values of 100, but uint8 FEE_PERC is using a multiplier 1e4. We recommend using the same multipliers for all variables that hold percent value. Line 46.

PreDex_Router.sol

1. `address predex_addr`, `address token_addr` variables are redundant. Lines 56, 60.
2. Gas usage in function open_prediction can be reduced. Computation is done twice. We recommend introducing a new local variable to hold the computation result. Lines 88, 90.

   Update: fixed in 6f34dc19c7ae83f863a7e91137eee170e7e0999c.
3. Typo in function documentation. Should be “Set close prediction allowance” Line 165.

   Update: fixed in 6f34dc19c7ae83f863a7e91137eee170e7e0999c.
4. We recommend adding a proxy-function for PreDexCore’s `function initiate_TWAP` for better user experience. With this function a user will interact with only one contract instead of two to open and close a prediction .
5. Wrong tenses in comment: “refund send back, part to POL pool, rest burnt”. Line 117.

   Update: fixed in 6f34dc19c7ae83f863a7e91137eee170e7e0999c.
6. `function set_predex_contract`, `function set_token_contract`, `function set_pool_addr` are missing checks for address params not to be zero addresses. Lines 134, 143,152.

   Update: fixed in 6f34dc19c7ae83f863a7e91137eee170e7e0999c.
7. `function set_reserve_param` is missing requirements checks for input parameter sanity. Line 184.
8. `function save_tokens` is redundant as its functionality is fully covered in `function save_erc20`. Line 191.

   Update: fixed in 6f34dc19c7ae83f863a7e91137eee170e7e0999c.
9. Variable `uint256 BETA_MAX_AMOUNT` should have the `constant` modifier as it cannot be changed. Line 49.

   Update: fixed in 6f34dc19c7ae83f863a7e91137eee170e7e0999c.

10. Variable `address POOL_ADDR` should use mixedCase as it can be changed by the owner of the contract. Line 53.

    Update: fixed in 6f34dc19c7ae83f863a7e91137eee170e7e0999c.

PreDex_Rewards.sol

1. SafeMath should be used for uint256 calculations. Lines 52, 64, 67, 70, 79, 87, 98, 100, 109.

   Update: some of the calculations changed to use SafeMath in ce3cff42951e7c7fdc849bc08f18e740437fff67.
2. Duplicate implementation of `function sqrt`. This function is already implemented in Babylonian contract. Line 153.
3. Variables `uint256 HOUR`, `uint256 WEEK`, `uint256 MONTH` are redundant as they are not used anywhere in the contract. Lines 14-16.

   Update: fixed in ce3cff42951e7c7fdc849bc08f18e740437fff67.
4. Variables `uint256 FACTOR` and `uint256 DECAY_START` should not use UPPER_CASE_WITH_UNDERSCORES style as this style is used for constants, but these variables can be changed by the contract owner. Recommendation: use mixedCase style for aforementioned variables. Lines 19-20.
5. `function set_reward_factor`, `function set_decay_start` are missing sanity checks for input parameters. Lines 171, 179.
6. Gas usage can be reduced in the `function calc_inv_perc`. Some of the computations can be precalculated. Line 79.

   Update: fixed in ce3cff42951e7c7fdc849bc08f18e740437fff67.

PreDEX_POL.sol

1. `function save_tokens` is redundant as its functionality is fully covered in `function save_erc20`. Line 189.

   Update: fixed in 41d74e43e198df8cb2dc5590da4df7198d498271.
2. Variable `address token_addr` is redundant. Line 42.
3. We recommend making `uint256 id`, `IERC20 pair` params in `event NewPool` and `event PoolEdited` indexed to make them searchable. Lines 45-46.
4. We recommend making `uint256 pool_id`, `address user` params in `event staked` and `event unstaked` indexed to make them searchable. Lines 47-48.

   Update: fixed in 41d74e43e198df8cb2dc5590da4df7198d498271.
5. `event staked` and `event unstaked` do not follow Solidity naming conventions. Lines 47-48.

Conclusion

No critical issues were found in the audited contracts. One high severity issue was found in the PreDex_POL contract. PRDX tokens can be stolen from this contract if a pool with malicious LP token is added to the contract. However the possibility of this attack is low as new pools are added only by the owner of the contract and it is considered that only trusted tokens will be added.

Several medium severity issues were found in the audited contracts.

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

Contracts have a lot of calculations but there are no tests in the repository. It is crucial to have 100% test coverage of the contracts functionality. We strongly recommend adding tests before going into the production to be sure that all the computations and contracts work as expected.

Update: the high severity issue and several medium and severity issues were fixed within the code update.