// HALBORN

Pontem Network - Liquidswap

Move Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **June 26th, 2022 - August 29th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY 3
CONTACTS 3
1 EXECUTIVE OVERVIEW 5
1.1 INTRODUCTION 6
1.2 AUDIT SUMMARY 6
1.3 TEST APPROACH & METHODOLOGY 6
RISK METHODOLOGY 7
1.4 SCOPE 9
2 ASSESSMENT SUMMARY & FINDINGS OVERVIEW 10
3 FINDINGS & TECH DETAILS 11
3.1 (HAL-01) REPEATED POOLS MAY BE CREATED - HIGH 13
Description 13
Code Location 14
Risk Level 14
Recommendation 14
Remediation plan 14
3.2 (HAL-02) ADMIN-ONLY SENSITIVE FUNCTIONS - LOW 15
Description 15
Code Location 15
Risk Level 17
Recommendation 17
Remediation plan 17
3.3 (HAL-03) LACK OF OWNERSHIP TRANSFER LOGIC - LOW 18
Description 18

Risk Level 18

Recommendation 18

Remediation plan 18

3.4 (HAL-04) OVERLY CENTRALIZED ACCOUNT PERMISSIONS - INFORMATIONAL 19

Description 19

Code Location 19

Risk Level 19

Recommendation 20

Remediation plan 20

3.5 (HAL-05) ABILITY TO CREATE CUSTOM LP TOKENS - INFORMATIONAL 21

Description 21

Code Location 21

Risk Level 21

Recommendation 22

Remediation plan 22

3.6 (HAL-06) MISSING EVENT EMISSION - INFORMATIONAL 23

Description 23

Code Location 23

Risk Level 24

Recommendation 24

Remediation plan 24

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	08/25/2022	Lukasz Mikula
0.2	Document Update	08/29/2022	Lukasz Mikula
0.3	Draft Review	09/05/2022	Luis Quispe Gonzales
0.4	Draft Review	09/06/2022	Gabi Urrutia
0.5	Document Update	09/06/2022	Lukasz Mikula
1.0	Remediation Plan	09/20/2022	Lukasz Mikula
1.1	Remediation Plan Review	09/20/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com

Luis Quispe Gonzales	Halborn	Luis.QuispeGonzales@halborn.com
Lukasz Mikula	Halborn	Lukasz.Mikula@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Pontem Network engaged Halborn to conduct a security audit on their smart contracts beginning on June 26th, 2022 and ending on August 29th, 2022. The security assessment was scoped to the smart contracts provided in the GitHub repository Pontem Network, commit hashes and further details can be found in the Scope section of this report.

1.2 AUDIT SUMMARY

The team at Halborn was provided five weeks for the engagement and assigned one full-time security engineer to audit the security of the smart contract. The security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit is to:

- Ensure that smart contract functions operate as intended
- Identify potential security issues with the smart contracts

In summary, Halborn identified some improvements to reduce the likelihood and impact of risks, which were mostly addressed by Pontem Network . The main ones are the following:

- Removed possibility to create repeated pools.
- Removed possibility to create unrestricted LP tokens.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual review of the code and automated security testing to balance efficiency, timeliness, practicality, and

accuracy in regard to the scope of the smart contract audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of smart contracts and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into the architecture, purpose, and use of the platform.
- Smart contract manual code review and walk-through to identify any logic issue.
- Thorough assessment of safety and usage of critical Rust variables and functions in scope that could lead to arithmetic related vulnerabilities.
- Test coverage review (aptos move test).
- On chain testing of core functions(aptos-cli)
- Deployment of Smart Contracts (Aptos Devnet)

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.
3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

Code repositories:

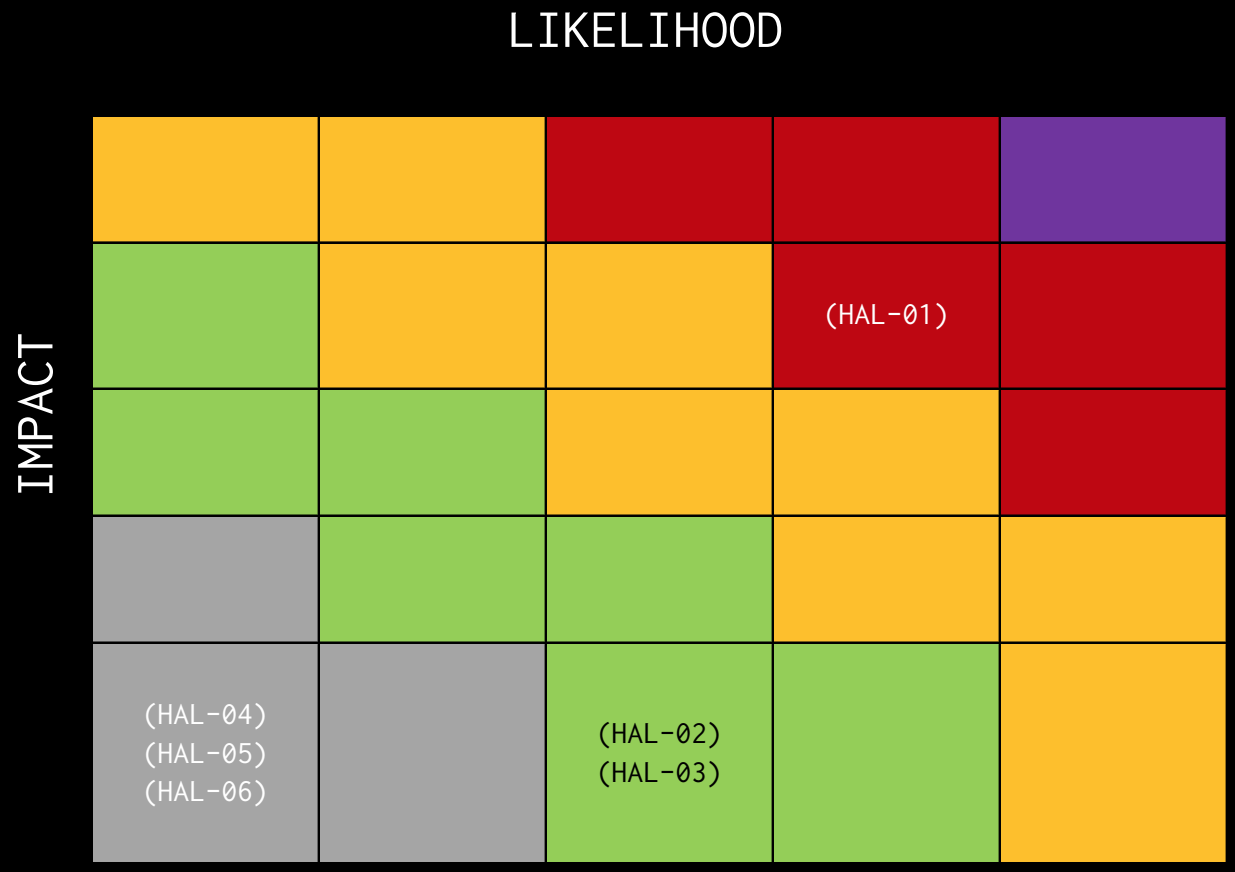
- https://github.com/pontem-network/uq64x64
- https://github.com/pontem-network/u256
- https://github.com/pontem-network/liquidswap-lp
- https://github.com/pontem-network/liquidswap

1. UQ64x64
   (a) Commit ID: f57d28041728ae36f0e5dc2164604f9f39856dae
   (b) Contracts in scope:
      - uq64x64.move
2. UQ256
   (a) Commit ID: 806718f699b2d0108368d1bd21b2f83cbad59b58
   (b) Contracts in scope:
      - u256.move
3. Liquidswap-lp
   (a) Commit ID: e49f05cabd67a29622d6f619df899d8b285ec393
   (b) Contracts in scope:
      - coins_extended.move
      - coins.move
      - lp.move
4. Liquidswap
   (a) Commit ID: 46bb75a71e1f43034493951880a9218dc0d378cc
   (b) Contracts in scope:
      - dao_storage.move
      - emergency.move
      - liquidity_pool.move
      - router.move
      - scripts.move

Out-of-scope: External libraries and financial related attacks.

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	1	0	2	3



EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - REPEATED POOLS MAY BE CREATED	High	SOLVED - 20/09/2022
HAL02 - ADMIN-ONLY SENSITIVE FUNCTIONS	Low	RISK ACCEPTED
HAL03 - LACK OF OWNERSHIP TRANSFER LOGIC	Low	RISK ACCEPTED
HAL04 - OVERLY CENTRALIZED ACCOUNT PERMISSIONS	Informational	ACKNOWLEDGED
HAL05 - ABILITY TO CREATE CUSTOM LP TOKENS	Informational	SOLVED - 20/09/2022
HAL06 - MISSING EVENT EMISSION	Informational	ACKNOWLEDGED

FINDINGS & TECH DETAILS

3.1 (HAL-01) REPEATED POOLS MAY BE CREATED - HIGH

Description:

It was observed that Aptos AMM allows for creating repeated pools. It should be noted, that these pools' LP will be different, which means that it is not possible to use LP's from one pool in another.

As a result, the liquidity in such pools might be low, which will negatively impact the resulting price of underlying assets. Since the aim of AMM protocol is to be competitive, the liquidity should be possibly maximal to offer most competitive prices to the users, which in turn encourages users to use the AMM. If certain coin pairs will be split over numerous pools, this will be difficult to achieve.

For instance, Astroport AMM disallows registering the same pools. Pancakeswap disallows repeated pools. Uniswap v3 allows restricted repeated pools - their number is limited, they have to differ in fees and are managed by governance.

Creating repeated pools is possible because upon creating liquidity pools the uniqueness check also considers LP token, and LP tokens can be arbitrarily chosen by the pool creator and do not have to be correlated with coins in the pool.
It should also be noted, that if at any time per-pool pausing will be introduced in the future, then this behavior can be used to circumvent trading restrictions on a pair.

For example, following pools created by address 0x999 can be created and will be treated as different assuming the coins that LP tokens consist of, are existing and registered under 0x999 account:

Listing 1

```
1 LiquidityPool<0x999::coin::BTC, 0x999::coin::USDT, 0x999::lp::<0
↳ x999::coins_extended::XYZ,0x999::coins_extended::ABC>>
2
```

```
3 LiquidityPool<0x999::coin::BTC, 0x999::coin::USDT, 0x999::lp::<0
↳ x999::coins_extended::ZXC,0x999::coins_extended::ZXC>>
```

Code Location:

Listing 2: liquidswap-0.2.5/sources/swap/scripts.move (Line 13)

```
10     /// Register a new liquidity pool for `X`/`Y` pair.
11     /// * `curve_type` - curve type: 1 = stable (like Solidly), 2
↳ = uncorrelated (like Uniswap).
12     public entry fun register_pool<X, Y, LP>(account: signer,
↳ curve_type: u8) {
13         router::register_pool<X, Y, LP>(&account, curve_type);
14     }
```

Risk Level:

Likelihood - 4
Impact - 4

Recommendation:

It might be worth considering disallowing repeated pools e.g., by allowing only LP tokens with coins they are, in fact, tied to. This is also explained in HAL-05.

Remediation plan:

SOLVED: The issue was solved in commit 1d4d3b6076f526b8fb6a95a9519f3930d67291a7. Pools are now non-repeating by design (note: different curve types are considered different pools), and it is no longer possible to create repeating pools using custom LP tokens, as LP tokens are created by the pool based on the coins it contains.

3.2 (HAL-02) ADMIN-ONLY SENSITIVE FUNCTIONS - LOW

Description:

Operations that are listed below are available to be or permanently disabling the project. According to the code, a single account, which is the deployer, can call them, which reduces decentralization. These functions are defined in dao_storage and emergency modules:

According to the code, a single account, which is the deployer, can call them, which reduces decentralization.

- dao_storage::withdraw (not callable at the moment of testing because there is no entry)
- emergency::pause
- emergency::resume
- emergency::disable_forever (not fully implemented at the moment of testing)

Below section presents exact code of these functions:

Code Location:

Listing 3: liquidswap-0.2.5/sources/swap/dao_storage.move (Lines 74,76)

```
74      public fun withdraw<X, Y, LP>(dao_admin_acc: &signer,
↳ pool_addr: address, x_val: u64, y_val: u64): (Coin<X>, Coin<Y>)
75      acquires Storage, EventsStore {
76          assert!(signer::address_of(dao_admin_acc) == @dao_admin,
↳ ERR_NOT_ADMIN_ACCOUNT);
77
78          let storage = borrow_global_mut<Storage<X, Y, LP>>(
↳ pool_addr);
79          let coin_x = coin::extract(&mut storage.coin_x, x_val);
80          let coin_y = coin::extract(&mut storage.coin_y, y_val);
81
```

```
82          let events_store = borrow_global_mut<EventsStore<X, Y, LP
↳ >>(pool_addr);
83          event::emit_event(
84              &mut events_store.coin_withdrawn_handle,
85              CoinWithdrawnEvent<X, Y, LP>{ x_val, y_val }
86          );
87
88          (coin_x, coin_y)
89      }
```

Listing 4: liquidswap-0.2.5/sources/swap/emergency.move (Lines 26,30)

```
26      public entry fun pause(account: &signer) {
27          assert!(!is_disabled(), ERR_DISABLED);
28          assert_no_emergency();
29
30          assert!(signer::address_of(account) == @emergency_admin,
↳ ERR_NO_PERMISSIONS);
31
32          move_to(account, Emergency {});
33      }
```

Listing 5: liquidswap-0.2.5/sources/swap/emergency.move (Lines 36,40)

```
35      /// Resumes all operations.
36      public entry fun resume(account: &signer) acquires Emergency {
37          assert!(!is_disabled(), ERR_DISABLED);
38
39          let account_addr = signer::address_of(account);
40          assert!(account_addr == @emergency_admin,
↳ ERR_NO_PERMISSIONS);
41          assert!(is_emergency(), ERR_NOT_EMERGENCY);
42
43          let Emergency {} = move_from<Emergency>(account_addr);
44      }
```

Listing 6: liquidswap-0.2.5/sources/swap/emergency.move (Lines 62,64)

```
61      /// Disable condition forever.
62      public entry fun disable_forever(account: &signer) {
63          assert!(!is_disabled(), ERR_DISABLED);
64          assert!(signer::address_of(account) == @emergency_admin,
```

```
↳ ERR_NO_PERMISSIONS);
65
66          move_to(account, Disabled {});
67      }
```

Risk Level:

Likelihood - 3
Impact - 1

Recommendation:

It should be noted that such sensitive operations should either be managed by a governance or a multi-signature wallet, to minimize risk of these methods being abused in case the single administrator account is compromised.

Remediation plan:

RISK ACCEPTED: The \client team accepted the risk of this finding.

3.3 (HAL-03) LACK OF OWNERSHIP TRANSFER LOGIC - LOW

Description:

It has been observed that no ownership transfer functionalities have been implemented in the contract, which means once contract is deployed under an account, this is the only administrator set for projects' lifetime. If the owner address is compromised, or if the development team needs to change the address for operational reasons, a significant portion of the contract's functionality will become unusable.

Risk Level:

Likelihood - 3
Impact - 1

Recommendation:

It is recommended to implement a two-step process where the owner nominates an account and the nominated account needs to call an acceptOwnership() function for the transfer of the ownership to fully succeed. This ensures the nominated account is a valid and active account.

Remediation plan:

RISK ACCEPTED: The \client team accepted the risk of this finding.

3.4 (HAL-04) OVERLY CENTRALIZED ACCOUNT PERMISSIONS - INFORMATIONAL

Description:

Liquidswap do not prevent one account to hold all privileged roles. Since these roles are pre-defined upon deployment, it is recommended to pay attention when deploying and do not give all important roles to just one user, which will increase overall decentralization.

Code Location:

The roles are defined in Move.toml:

Listing 7: liquidswap-0.2.5/Move.toml (Lines 6-8)

```
1 [package]
2 name = "Liquidswap"
3 version = "0.2.3"
4
5 [addresses]
6 liquidswap = "7
↳ E05770F81CB187C4EDB4D7047D0C53025E3CDEAB33215627E5609E2A325FCEF"
7 dao_admin = "7
↳ E05770F81CB187C4EDB4D7047D0C53025E3CDEAB33215627E5609E2A325FCEF"
8 emergency_admin = "7
↳ E05770F81CB187C4EDB4D7047D0C53025E3CDEAB33215627E5609E2A325FCEF"
9
```

If the contract is deployed in that state and the only account is compromised, then the attacker will have full control over projects and the funds accumulated in it.

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

It is recommended to use different addresses for these roles/modules upon deployment.

Remediation plan:

ACKNOWLEDGED: The \client team acknowledged this finding.

3.5 (HAL-05) ABILITY TO CREATE CUSTOM LP TOKENS - INFORMATIONAL

Description:

It has been observed that each pool takes as an argument LP token for registering. These LP tokens are specified at pool creation and can be uncorrelated with assets that are present in the pool. For example, a BTC/USDT pool might use exemplary LP `0xCreator::lp::<0xANY::coins_extended::X,0xANY::coins_extended::Y>`.

This behavior has two impacts: first, it allows for creating repeated pools, as in HAL-01.
Second case, it reduces transparency and causes confusion, since the LP's held on an account are not correlated to assets they are tied to.

It is often noted that AMM protocols' pools issue LP tokens based on the coins supplied to the pools. In such case, the LP argument would not be taken for a pool registration, and the pool could create the token itself based on what coins were supplied to the pool.

Code Location:

Listing 8: liquidswap-0.2.5/sources/swap/scripts.move (Line 13)

```
10      /// Register a new liquidity pool for `X`/`Y` pair.
11      /// * `curve_type` - curve type: 1 = stable (like Solidly), 2
↳ = uncorrelated (like Uniswap).
12      public entry fun register_pool<X, Y, LP>(account: signer,
↳ curve_type: u8) {
13          router::register_pool<X, Y, LP>(&account, curve_type);
14      }
```

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

It is recommended to consider removing the possibility to specify the token at pool registration and delegate that functionality to the pool itself, creating a LP token which consists of coins from the pool. This would also mitigate HAL-01 - as the pools will always be created with predictable LP token, there won't be possibility to create repeated pools with custom LP tokens.

Remediation plan:

SOLVED: The issue was solved in commit 1d4d3b6076f526b8fb6a95a9519f3930d67291a7 . LP tokens are now created based on the actual coins in the pool.

3.6 (HAL-06) MISSING EVENT EMISSION - INFORMATIONAL

Description:

It has been observed that key administrative functions that modify project state do not emit events upon being used. This decreases visibility of key events in the blockchain. The below functions, all belonging to emergency module, do not emit events upon being called.

Code Location:

Listing 9: liquidswap-0.2.5/sources/swap/emergency.move (Line 32)

```
26     public entry fun pause(account: &signer) {
27         assert!(!is_disabled(), ERR_DISABLED);
28         assert_no_emergency();
29
30         assert!(signer::address_of(account) == @emergency_admin,
↳ ERR_NO_PERMISSIONS);
31
32         move_to(account, Emergency {});
33     }
```

Listing 10: liquidswap-0.2.5/sources/swap/emergency.move (Line 43)

```
35     /// Resumes all operations.
36     public entry fun resume(account: &signer) acquires Emergency {
37         assert!(!is_disabled(), ERR_DISABLED);
38
39         let account_addr = signer::address_of(account);
40         assert!(account_addr == @emergency_admin,
↳ ERR_NO_PERMISSIONS);
41         assert!(is_emergency(), ERR_NOT_EMERGENCY);
42
43         let Emergency {} = move_from<Emergency>(account_addr);
44     }
```

Listing 11: liquidswap-0.2.5/sources/swap/emergency.move (Line 66)

```
61      /// Disable condition forever.
62      public entry fun disable_forever(account: &signer) {
63          assert!(!is_disabled(), ERR_DISABLED);
64          assert!(signer::address_of(account) == @emergency_admin,
↳ ERR_NO_PERMISSIONS);
65
66          move_to(account, Disabled {});
67      }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to implement event emission on operations such as Pause, Resume or Disable.

Remediation plan:

ACKNOWLEDGED: The \client team acknowledged this finding.

THANK YOU FOR CHOOSING

// HALBORN