Local Money Audit Report

Prepared 19th January, 2023

Table of Contents

Introduction

The LocalMoney team have engaged with A10 Capital to provide a full security audit of their CosmWasm smart contracts located at

Repo: https://github.com/Local-Money/localmoney/tree/main/contracts/contracts

Commit: 7197ae33fc5c77d2e9af1e8f648c89b093c2c843

Methodology

As part of the review process, we have checked that the code:

- Has documentation and comments that match the logic and behaviour
- Is not affected by any known security vulnerabilities

Our team followed best practices & industry-standard techniques to verify the proper implementation of the smart contract. Our security auditing experts have reviewed the contract line-by-line and documented any issues as when they were discovered.

The review that has taken place on the codebase, included the following:

1. Due diligence during assessment of overall code quality of the codebase
2. Testing contract logic against common & uncommon attack vectors
3. Thorough, manual review of the codebase, line-by-line

In addition to the manual code review, we also engaged using code fuzzers & monitored for exceptions, or failures through built in code assertions.

Summary of Findings

Impact	Summary	Location	Status
HIGH	Critical Parameters of Hub Can be Modified Unrestricted	hub/src/contract.rs - Line 53	Resolved
HIGH	Offers Potential denial of service	offer/src/contract.rs	Resolved
HIGH	Trade Potential denial of service	trade/src/contract.rs	Resolved
MEDIUM	Inadequate handling of uppercase addresses	hub/src/contract.rs - Line 105	Acknowledged
MEDIUM	Privileged address can be transferred without confirmation	hub/src/contract.rs - Line 105	
MEDIUM	Prices from oracle can become stale	price/src/contract.rs - Line 77	
MEDIUM	Weak source of randomness for arbitrator selection	trade/src/contract.rs - Line 238	
MEDIUM	Missing validation mechanism for offer parameters	offer/src/contract.rs - Line 80	
MEDIUM	Denom is lost when sending more than one during fund	trade/src/contract.rs - Line 426	
LOW	Usage of magic numbers in the contract	trade/src/contract.rs - Line 168 price/src/contract.rs - Line 136 price/src/contract.rs - Line 162	
LOW	Lose Precision When Calculating Prices	price/src/contract.rs	
LOW	Overflow set only for profile	N/A	
INFO	Usage of deprecated crates	N/A	
INFO	Incomplete documentation	N/A	
INFO	Incomplete test cases	N/A	

High - Critical Parameters of Hub Can Be Modified Unrestricted

Description

Incorrect use of the `update_config` function in contracts/hub/src/contract.rs can modify the critical parameters to invalid values and inadvertently. As a consequence, this could lead to unfavorable transactions.

Code Location

https://github.com/Local-Money/localmoney/tree/7197ae33fc5c77d2e9af1e8f648c89b093c2c843/contracts/contracts/hub/src/contract.rs#L53

Recommendation

Update the logic of `update_config` function to include a MINIMAL and MAXIMAL threshold values for the following parameters:

- burn_fee_pct
- chain_fee_pct
- warchest_fee_pct
- trade_expiration_timer
- trade_dispute_timer

Remediation

The issue has been fixed in commit 537f5cf2b8849f30cf113449f96df975233fac6a

High – Offers Potential Denial of Service

Description

The current implementation of the `contracts/contracts/offer/src/contract.rs` contract does not have a mechanism to remove archive offers. Consequently, this could lead to unnecessary Storage usage and denial of service.

Attack Scenario

1. Alice posts an offer
2. Alice archives the offer
3. Alice creates new offer
4. Situation repeats itself n-times

Proof of Concept

Below we try to read the last offer of the contract, instead, we will get the 99th offer.

Print user offers count:

```
{ "count": 256 }
```

Print last offer, limited by 1:

```
[
  {
    offer: {
      id: '97_99',
      owner: 'osmo18s5lynnmx37hq4wlrw9gdn68sg2uxp5rgk26vv',
      offer_type: 'buy',
      fiat_currency: 'USD',
      rate: '97',
      min_amount: '100',
      max_amount: '1000',
      description: null,
      denom: [Object],
      state: 'archive',
      timestamp: 1672149414
    },
    profile: {
      addr: 'osmo18s5lynnmx37hq4wlrw9gdn68sg2uxp5rgk26vv',
      created_at: 1672148427,
      requested_trades_count: 0,
      released_trades_count: 0,
      last_trade: 0,
```

```
      contact: 'maker001',
      encryption_key:
  'MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA2vDhS/v72mP4xiONjaH+m7t6IZ5Eaw0hwkacKz
  42kSMOx7TDl1r+xAccIrRnjHMA87GZr1pRboZAvIgHO9/adjPMlXxwU8dVfnaBfpXUQoGfukvacWDuu5khc
  dbuCzH7NnI5F2gDy6i3fwQ1ls5kc94vHlomaZ0+HGFqPT48SBgWMSsBFxrCCAL3uvL7VIl5OQUiFQb6mPE3
  grPO4M/Q6gKeZYoT3xzUb00XhQZF+i6EvYThVPAqYM4mwyPApaSLIo2SMeaHgS0kBRiyBM2vTv5QYf6C5Pb
  5hsOuprffNBPyvCIwPuDuIe3rAH8yI67FKTxn5qK3UWEk8B7StI01DwIDAQAB',
      active_offers_count: 2,
      active_trades_count: 0
    }
  }
]
```

Below we try to read the last 100 offers of the user, instead, we will receive an error:

Print last 100 offers fail:

```
(node:15554) UnhandledPromiseRejectionWarning: Error: Query failed with (18): out of gas in location: wasm contract; gasWanted: 3000000, gasUsed: 3008680: out of gas: invalid request
```

Recommendation

It is recommended to implement a mechanism to remove unnecessary offers from the contract to avoid unnecessary Storage usage and avoid denial of service when querying the offers.

Remediation

The issue has been fixed in commit [c15f78a8470ee951fa4a503cf7bc3415623e1065](c15f78a8470ee951fa4a503cf7bc3415623e1065).

High – Trades Potential Denial of Service

Description

The current implementation of the `contracts/contracts/trades/src/contract.rs` contract does not have a mechanism to remove archive offers. Consequently, this could lead to unnecessary Storage usage and denial of service.

Attack Scenario

1. Alice posts an offer
2. Bob creates a trade
3. Alice cancels the trade
4. Bob creates trade again
5. Situation repeats itself n-times

Proof of Concept

Below we try to read the last 10 trades of the user. Instead, we will receive an error:

Print Current amount of Trades:

```
(node:20136) UnhandledPromiseRejectionWarning: Error: Query failed with (18): out of gas in location: wasm contract; gasWanted: 3000000, gasUsed: 3004335: out of gas: invalid request
```

Recommendation

It is recommended to implement a way to remove unnecessary trades from the contract to avoid unnecessary Storage usage and avoid denial of service when querying the offer.

Remediation

The issue has been fixed in commit 7b1790203d19ecb5c31524f97c78dd9e26cf7965.

Medium – Inadequate Handling of Uppercase Addresses

Description

The Local Money protocol does not consider that addresses may be valid in both upper and lower case. The Bech32 encoding uses an alphabet composed of 32 letters and numbers and allows addresses to be valid if all letters are lowercase or uppercase, however a strict comparison between them will not succeed.

Lack of address normalization in CosmWasm contracts can cause numerous undesirable effects.

Code Location

https://github.com/Local-Money/localmoney/tree/7197ae33fc5c77d2e9af1e8f648c89b093c2c843/contracts/contracts/hub/src/contract.rs#L105

Recommendation

There are two approaches that can be taken to address the issue of strict comparison between upper and lower case addresses:

1. Update the cosmwasm- vm and use the cosmwasm_std::Api::addr_validate function, as described in reference CWA-2022-002.
2. If the update is not possible, store addresses in canonical format using the cosmwasm_std::Api::addr_canonicalize utility function.

When using the second option, keep the following in mind:

- To compare canonicalized addresses, both addresses must be in canonical format. For example, when performing access controls, the sender's address (e.g. info.sender or env.message.sender) should also be canonicalized.
- To send funds to a canonicalized address or include it in a message to a different contract, it must first be converted to its human-readable format using the cosmwasm_std::Api::addr_humanize utility function.

Medium – Privileged Address Can Be Transferred Without Confirmation

Description

An incorrect use of the `update_admin` function in **contracts/hub/src/contract.rs** can set the owner to an invalid address and inadvertently lose control of the contracts, which cannot be undone in any way.

Currently, the owner of the contracts can change **owner address** using the function in a `single transaction` and `without confirmation` from the new address.

Code Location

Appendix A

https://github.com/Local-Money/localmoney/tree/7197ae33fc5c77d2e9af1e8f648c89b093c2c843/contracts/contracts/hub/src/contract.rs#L105

Recommendation

It is recommended to split the owner transfer functionality into `set_owner` and `accept_ownership` functions. The latter function allows the transfer to be completed by the recipient.

Medium – Prices From Oracle Can Become Stale

Description

`update_prices` function in **contracts/price/src/contract.rs** allows `price_provider_addr` to feed the contract with the prices. `query_fiat_price_for_denom` function allows a user to query the contract for the fiat price for a given `denom`, however, the function does not previously validate if the price has been updated within a reasonable timeframe.

Consequently, `fiat` prices can become stale if `price_provider_addr` does not feed prices enough or if the off-chain trigger does not work correctly (out-of-scope for this audit), which could negatively affect users' operations or protocol funds.

Code Location

Appendix B

https://github.com/Local-Money/localmoney/tree/7197ae33fc5c77d2e9af1e8f648c89b093c2c843/contracts/contracts/price/src/contract.rs#L77

Recommendation

It is recommended to apply one of the following oracle strategies:

- Update the logic of `query_fiat_price_for_denom` function to throw an error message if price has not been updated within a reasonable timeframe defined in the contract.

- If protocol prioritization is placed on data freshness, an oracle using moving averages can be utilized instead of only measuring the cumulative price variable once per period. For further information on this approach, refer to the following:

https://docs.uniswap.org/protocol/V2/guides/smart-contract-integration/building-an-oracle#moving-averages

Medium – Weak Source of Randomness for Arbitrator Selection

Description

`create_trade` function allows the user to create a trade for the selected offer, during the creation of the trade an arbitrator is appointed. The implementation of the protocol suggests that the arbitrator is chosen randomly. However current implementation is:

- Using a weak source of randomness which could be easily predicted
- Arbiter can be same person as seller or buyer
- Same arbiter can be registered **X** times for a given denom
- There can be only one arbitrator registered

Code Location

https://github.com/Local-Money/localmoney/tree/7197ae33fc5c77d2e9af1e8f648c89b093c2c843/contracts/contracts/trade/src/contract.rs#L238

```
1.     let random_seed: u32 = (env.block.time.seconds() % 100) as u32;
2.      let arbitrator = ArbitratorModel::get_arbitrator_random(
3.          deps.as_ref(),
4.          random_seed as usize,
5.          offer.fiat_currency.clone(),
6.      );
```

Recommendation

It is recommended to use a reliable source of randomness for arbitrator selection, integration such as the Entropy Beacon on Kujira or an off-chain oracle such as ChainLink for example.

Moreover, it is recommended to throw an appropriate error if there are not enough arbitrators in the protocol or arbitrator is buyer or seller.

Medium – Missing Validation Mechanism for Offer Parameters

Description

`create offer` allow the user to create a new offer in the protocol, however current implementation does not validate the correctness of the given parameters.

Code Location

Appendix C

https://github.com/Local-Money/localmoney/tree/7197ae33fc5c77d2e9af1e8f648c89b093c2c843/contracts/contracts/offer/src/contract.rs#L80

Recommendation

It is recommended to validate all input parameters in the `create_offer` function. Consider implementation of validation mechanism which check is denom is not empty and `rate` value is in range of safe MINIMUM and MAXIMUM values.

Medium – Denom Is Lost When Sending More Than One During Fund

Description

The `fund_escrow` function does not perform a basic check to detect if more than one denom has been sent during the deposit. In that case, the additional denom sent in this way would be locked in the contract.

Code Location

Appendix D

https://github.com/Local-Money/localmoney/tree/7197ae33fc5c77d2e9af1e8f648c89b093c2c843/contracts/contracts/trade/src/contract.rs#L426

Recommendation

It is recommended to throw an appropriate error when the user sends more than one denom. Consider the implementation of an appropiate validation check for denoms that have been sent to the contract.

Low – Usage of Magic Numbers in the Contract

Description

Current implementation of the protocol uses `MAGIC NUMBERS`.

Code Location

https://github.com/Local-Money/localmoney/tree/7197ae33fc5c77d2e9af1e8f648c89b093c2c843/contracts/contracts/trade/src/contract.rs#L168

https://github.com/Local-Money/localmoney/tree/7197ae33fc5c77d2e9af1e8f648c89b093c2c843/contracts/contracts/price/src/contract.rs#L136

https://github.com/Local-Money/localmoney/tree/7197ae33fc5c77d2e9af1e8f648c89b093c2c843/contracts/contracts/price/src/contract.rs#L162

Recommendation

It is recommended to wrap `magic numbers` into constant variables with appropriate names.

Low – Lose Precision When Calculating Prices

Description

It been observed that calculations in `contracts/contracts/price/src/contract.rs` use 12 decimal places instead of 18.

Recommendation

It is recommended using a `DECIMAL_FRACTIONAL` of 1_000_000_000_000_000_000 as CosmWasm does.

Low – Overflow Set Only for Profile

Description

It has been observed that only the workspace cargo.toml file has enabled overflow checks for the release profile. The individual packages have not enabled release overflow checks.

Recommendation

Even though this check is automatically applied to all packages listed in the workspace's cargo.toml file, it is advisable to also explicitly enable overflow checks for each individual package. This can prevent unintended consequences during refactoring of the project.

Info – Usage of Deprecated Crates

Description

```
Crate:     parity-wasm
Version:   0.42.2
Warning:   unmaintained
Title:     Crate `parity-wasm` deprecated by the author
Date:      2022-10-01
ID:        RUSTSEC-2022-0061
URL:       https://rustsec.org/advisories/RUSTSEC-2022-0061
Dependency tree:
parity-wasm 0.42.2
└── cosmwasm-vm 1.0.0
    ├── trade 0.0.0
    ├── profile 0.0.0
    ├── price 0.0.0
    ├── offer 0.0.0
    ├── localmoney-protocol 1.0.0
    │   ├── trade 0.0.0
    │   ├── profile 0.0.0
    │   ├── price 0.0.0
    │   ├── offer 0.0.0
    │   └── hub 0.0.0
    └── hub 0.0.0

Crate:     crossbeam-utils
Version:   0.8.5
Warning:   yanked
Dependency tree:
crossbeam-utils 0.8.5
├── rayon-core 1.9.1
│   └── rayon 1.5.1
│       ├── wasmer-compiler-singlepass 2.2.1
│       │   └── wasmer 2.2.1
│       │       ├── wasmer-middlewares 2.2.1
│       │       │   └── cosmwasm-vm 1.0.0
│       │       │       ├── trade 0.0.0
│       │       │       ├── profile 0.0.0
│       │       │       ├── price 0.0.0
│       │       │       ├── offer 0.0.0
│       │       │       ├── localmoney-protocol 1.0.0
│       │       │       │   ├── trade 0.0.0
│       │       │       │   ├── profile 0.0.0
```

```
│       │       │       │   ├── price 0.0.0
│       │       │       │   ├── offer 0.0.0
│       │       │       │   └── hub 0.0.0
│       │       │       └── hub 0.0.0
│       │       └── cosmwasm-vm 1.0.0
│       └── wasmer-compiler-cranelift 2.2.1
│           └── wasmer 2.2.1
├── crossbeam-epoch 0.9.5
│   └── crossbeam-deque 0.8.1
│       ├── rayon-core 1.9.1
│       └── rayon 1.5.1
├── crossbeam-deque 0.8.1
└── crossbeam-channel 0.5.1
    └── rayon-core 1.9.1
```

Info – Incomplete Documentation

Description

The documentation provided by localmoney is incomplete. For instance, the documentation included in the GitHub repository should include a contract diagram, instructions for users on how to interact with the contracts, list of the contracts with usage purpose and a walkthrough on how to deploy and test the smart contracts.

Recommendation

Consider updating the documentation in GitHub to clarify data flow, user usage and to enable greater ease when contracts are deployed and tested. Have a non-developer or QA resource work through the process to make sure it addresses any gaps in the set-up steps due to technical assumptions.

Info – Incomplete Test Cases

Description

The codebase that was submitted has a smaller number of integration tests spread across multiple libraries.

Recommendation

It is advisable to expand the coverage of the tests, particularly to include extreme cases that may not be easily identified during a manual review.

Code Appendix

Appendix A – Privileged Address Can Be Transferred Without Confirmation

```
1. fn update_admin(
2.     deps: DepsMut,
3.     info: MessageInfo,
4.     new_admin: Addr,
5. ) -> Result<Response, ContractError> {
6.     let mut admin = ADMIN.load(deps.storage).unwrap();
7.     if !info.sender.eq(&admin.addr) {
8.         return Err(Unauthorized {
9.             owner: admin.addr.clone(),
10.            caller: info.sender.clone(),
11.        });
12.    }
13.
14.    let old_admin = admin.addr.clone();
15.    admin.addr = new_admin.clone();
16.    ADMIN.save(deps.storage, &admin).unwrap();
17.
18.    let res = Response::new()
19.        .add_attribute("action", "update_admin")
20.        .add_attribute("old_admin", old_admin)
21.        .add_attribute("new_admin", new_admin);
22.    Ok(res)
23.}
```

Link to Github

Appendix B – Prices from Oracle Can Become Stale

```
1. pub fn update_prices(
2.     deps: DepsMut<KujiraQuery>,
3.     env: Env,
4.     info: MessageInfo,
5.     prices: Vec<CurrencyPrice>,
6. ) -> Result<Response<KujiraMsg>, ContractError> {
7.     let hub_cfg = get_hub_config(deps.as_ref());
8.     assert_ownership(info.sender, hub_cfg.price_provider_addr)?;
9.     let mut attrs: Vec<(&str, String)> = vec![("action",
   "update_prices".to_string())];
10.    prices.iter().for_each(|price| {
11.        // Load existing object or default
12.        let path = FIAT_PRICE.key(price.currency.to_string().as_str());
13.        let mut currency_price = path
14.            .load(deps.storage)
15.            .unwrap_or(CurrencyPrice::new(price.currency.clone()));
16.
17.        // Update price
18.        currency_price.usd_price = price.usd_price;
19.        currency_price.updated_at = env.block.time.seconds();
20.        path.save(deps.storage, &currency_price).unwrap();
21.        attrs.push(("currency", price.currency.to_string()));
22.        attrs.push(("usd_price", price.usd_price.to_string()));
23.    });
24.    let res = Response::new().add_attributes(attrs);
25.    Ok(res)
```

Link to Github

Appendix C – Missing Validation Mechanism for Offer Parameters

```
1. pub fn create_offer(
2.     deps: DepsMut,
3.     env: Env,
4.     info: MessageInfo,
5.     msg: OfferMsg,
6. ) -> Result<Response, ContractError> {
7.     let hub_config = get_hub_config(deps.as_ref());
8.     assert_min_g_max(msg.min_amount, msg.max_amount)?;
9.
10.    assert_offer_description_valid(msg.description.clone()).unwrap();
11.
12.    // Load offers count to create the next sequential id, maybe we can
   switch to a hash based id in the future.
13.    let mut offers_count = offers_count_storage(deps.storage)
14.        .load()
15.        .unwrap_or(OffersCount { count: 0 });
16.    offers_count.count += 1;
17.    let offer_id = [msg.rate.clone().to_string(),
   offers_count.count.to_string()].join("_");
18.
19.    // Update profile contact info
20.    let update_profile_contact_msg = update_profile_contact_msg(
21.        hub_config.profile_addr.to_string(),
22.        info.sender.clone(),
23.        msg.owner_contact.clone(),
24.        msg.owner_encryption_key.clone(),
25.    );
26.
27.    let offer = OfferModel::create(
28.        deps.storage,
29.        Offer {
30.            id: offer_id,
31.            owner: info.sender.clone(),
32.            offer_type: msg.offer_type,
33.            fiat_currency: msg.fiat_currency.clone(),
34.            rate: msg.rate,
35.            denom: msg.denom,
36.            min_amount: msg.min_amount,
37.            max_amount: msg.max_amount,
38.            state: OfferState::Active,
```

```
            description: msg.description,
            timestamp: env.block.time.seconds(),
        },
    )
    .offer;

    // Update offers count
    offers_count_storage(deps.storage)
        .save(&offers_count)
        .unwrap();

    // Update profile active offers
    let update_profile_offers_msg = update_profile_active_offers_msg(
        hub_config.profile_addr.to_string(),
        info.sender.clone(),
        offer.state,
    );

    let res = Response::new()
        .add_submessage(update_profile_contact_msg)
        .add_submessage(update_profile_offers_msg)
        .add_attribute("action", "create_offer")
        .add_attribute("type", offer.offer_type.to_string())
        .add_attribute("id", offer.id.to_string())
        .add_attribute("rate", offer.rate.to_string())
        .add_attribute("min_amount", offer.min_amount.to_string())
        .add_attribute("max_amount", offer.max_amount.to_string())
        .add_attribute("owner", offer.owner);
    Ok(res)
}
```

Link to Github

Appendix D – Denom is Lost When Sending More Than One During Fund

```
1. fn fund_escrow(
2.     deps: DepsMut,
3.     env: Env,
4.     info: MessageInfo,
5.     trade_id: String,
6.     maker_contact: Option<String>,
7. ) -> Result<Response, ContractError> {
8.     // Load HubConfig, Trade & Offer
9.     let hub_config = get_hub_config(deps.as_ref());
10.    let mut trade = TradeModel::from_store(deps.storage, &trade_id);
11.    let offer = load_offer(
12.        &deps.querier.clone(),
13.        trade.offer_id.clone(),
14.        trade.offer_contract.to_string(),
15.    )
16.    .unwrap()
17.    .offer;
18.
19.    // Ensure the message has the correct funds
20.    let trade_denom = &denom_to_string(&trade.denom);
21.    let balance = match info.funds.first().unwrap() {
22.        coin if coin.denom.eq(trade_denom) => coin.clone(),
23.        _ => {
24.            let received =
   info.funds.first().unwrap_or(&Coin::default()).denom.clone();
25.            return Err(InvalidDenom {
26.                expected: trade_denom.clone(),
27.                received,
28.            });
29.        }
30.
```

Link to Github