SECURITY AUDIT OF

REMITANO ORACLE AND PEGGED-FIAT PROGRAMS

Public Report

Jul 31, 2023

Verichains Lab

info@verichains.io

https://www.verichains.io

Driving Technology > Forward

ABBREVIATIONS

Name	Description
RENEC	A decentralized blockchain built to enable scalable, user-friendly apps for the world.
RENEC	A cryptocurrency whose blockchain is generated by the RENEC platform.
Lamport	A fractional native token with the value of 0.000000001 RENEC.
Program	An app interacts with a RENEC cluster by sending it transactions with one or more instructions. The RENEC runtime passes those instructions to program.
Instruction	The smallest contiguous unit of execution logic in a program.
Cross-program invocation (CPI)	A call from one smart contract program to another.
Anchor	A framework for RENEC's Sealevel runtime providing several convenient developer tools for writing smart contracts.

EXECUTIVE SUMMARY

This Security Audit Report was prepared by Verichains Lab on Jul 31, 2023. We would like to thank the Remitano for trusting Verichains Lab in auditing smart contracts. Delivering high-quality audits is always our top priority.

This audit focused on identifying security flaws in code and the design of the Remitano Oracle and Pegged-Fiat programs. The scope of the audit is limited to the source code files provided to Verichains. Verichains Lab completed the assessment using manual, static, and dynamic analysis techniques.

During the audit process, the audit team had identified some vulnerable issues in the smart contracts code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About Remitano Oracle and Pegged-Fiat programs

Remitano Network is designed with extendable awareness from beginning. The network is designed so that it is possible to add new capabilities and leverage the latest exciting technology in the blockchain space to the network without the need to introduce a new token.

Remitano Oracle and Pegged-Fiat programs is a system that allows users to lock their fiat-based tokens to mint other tokens based on prices provided by `Oracle` program.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the Remitano Oracle and Pegged-Fiat programs. The scope of the audit is limited to the source code files provided to Verichains.

SHA256 Sum	File
6b0829aff39282c7925f4ab97d74f78d85ae787be63e8b3bf0db5a4d52e11465	remitano.zip

The latest version of the following file was made available in the course of the review:

SHA256 Sum	File
c6744b90e866b87e9a549a1006e5beb0cb0306d61c0e2645ccf53bf31b550e0d	pegged_fiat-master/programs/pegged-fiat/src/instructions/lock_n_mint.rs
a0ff130416a991ad8b147c3a3ec1e2ada0ce61005c69e6b2aae9d9e22102ab95	pegged_fiat-master/programs/pegged-fiat/src/instructions/initialize.rs
701fce8200c88e38e779c0e00a18764d6cf3f054d07fa073cd046a8cf023d07c	pegged_fiat-master/programs/pegged-fiat/src/instructions/mod.rs
6ffd68a9d48fabbbc90cfd5e5f620d46ccee84fa8c58b70f03085a0076c7fa58	pegged_fiat-master/programs/pegged-fiat/src/instructions/on_off_pair.rs
563c5650b066281378fdd7f8a67a9d99debb38b58a270d280230a023a9f06939	pegged_fiat-master/programs/pegged-fiat/src/instructions/add_pair.rs
bab3f5a52b0f755c8ddf74f341ffa8975a5a3a46593a1cd0dfcd83854e25cdc2	pegged_fiat-master/programs/pegged-fiat/src/constants.rs
82cbf4bd5431230b07d043cf696aa83afe0af14da891f49e98bbf84c16fbb43eS	pegged_fiat-master/programs/pegged-fiat/src/util/util.rs

424ed95f83f33c29cded8c87f62c8b561dd1e0c633624 13af4b6c946d9ade09d	pegged_fiat-master/programs/pegged-fiat/src/util/token.rs
c3c3a6c9bd2778bceca72a0ac269e1924126de50c54cc e46ae13dbc41915b6ad	pegged_fiat-master/programs/pegged-fiat/src/util/mod.rs
85980c68185e0fb301eaa150c1f48aa1431facea32400 47d95c37c67ed71cb7f	pegged_fiat-master/programs/pegged-fiat/src/lib.rs
c43a465467b7eeb97f0745bca55d42a585ba4c0b076cb e15196e854a91107500	pegged_fiat-master/programs/pegged-fiat/src/state/controller.rs
ad2d685ecbea469fcc6d5e74f747b27ede063cf574074 fbfd86bd7b4818ddcc3	pegged_fiat-master/programs/pegged-fiat/src/state/pair.rs
aaa3e4c990a76d350a687690b2b94b0ec1e3c44e0de6c e9d72a9cb6c381341fc	pegged_fiat-master/programs/pegged-fiat/src/state/mod.rs
82ba4aab98601c7da0c849b55e5a907ddcffe950d405e 245ae60699b124213ba	pegged_fiat-master/programs/pegged-fiat/src/state/user_profile.rs
32853f169ed8a4fbe3c40a3f407c84943627e33a44d45 1ca377e9d6406faf6f0	pegged_fiat-master/programs/pegged-fiat/src/state/lock_profile.rs
19661e5aaceb7e1e91063698d2f108036ba14a1f3ba7b 6aaeaf3a67adac1c017	pegged_fiat-master/programs/pegged-fiat/src/state/oracle_states.rs
47c9d8f4d4464cd42a1c7c0b3ae660d8ff4315af440cb e8152128d0b99c6b764	pegged_fiat-master/programs/pegged-fiat/src/errors.rs
db668cfb3a26c65a120ea511dd4917fb73119850a2f25 4d3b03ee302213698d9	oracle-program-master/programs/oracle/src/instructions/add_publisher.rs
3e87b249ed9217d2bceaa6dd2545417e100506a4fdda9 acf01798035c4078a9a	oracle-program-master/programs/oracle/src/instructions/add_product.rs
15157d837fed75ed091e96efa17ce930330f66838d7c4 2fb669282b0fc735b94	oracle-program-master/programs/oracle/src/instructions/remove_publisher.rs
e065cf175eff2c67cba0825f72244d117fa0cddbdd607 4a58d276c0392122bd9	oracle-program-master/programs/oracle/src/instructions/set_safe_range.rs
f4bd0ffb9942a4319a8590cc2e8923a52f57f0f3b9695 890136211bc69dd35e2	oracle-program-master/programs/oracle/src/instructions/initialize.rs

ed01691679ce03ffee8f27aa0b9173ce750ace34c26bc67f4a0586abfa5589f3	oracle-program-master/programs/oracle/src/instructions/mod.rs
6abf03ede0cb74c09caf8eadb1f78288808a1363218783a6cd73557d20d422f3	oracle-program-master/programs/oracle/src/instructions/post_price.rs
49706ba30a6733a9d4439aab971776376b42b0007f3cd6430b9862c635237001	oracle-program-master/programs/oracle/src/constants.rs
82cbf4bd5431230b07d043cf696aa83afe0af14da891f49e98bbf84c16fbb43e	oracle-program-master/programs/oracle/src/util/util.rs
41f31437869bc740529de994e79808ca80eb4fc9522329b67660645b9f4a2dd9	oracle-program-master/programs/oracle/src/util/mod.rs
b0a74aa37bf58208823cd0acd01c26b27b42af2dcc5055837e20a24649deb21b	oracle-program-master/programs/oracle/src/lib.rs
46bb5ff3e5c67783ad253bfccff7d6dab0bf6696b3afd9e8404d098d7d7397f5	oracle-program-master/programs/oracle/src/state/controller.rs
1f7a2240dd3e2c80ebfb06bbebe4bbe6edc204cb0644c601139dcba235f7b317	oracle-program-master/programs/oracle/src/state/mod.rs
fef2605c61409bb41d39c819dcc0c3196c3f374931434bd99d908b6616d7f698	oracle-program-master/programs/oracle/src/state/price.rs
84f9bf4c064ed69a0937194da987c0209c9702ff2c836757a8a1ea5ffb04dc7f	oracle-program-master/programs/oracle/src/state/publisher.rs
9973928ac2a30c3415d210d58d0c5711dca6be68aca44e5262bebde4173bdda5	oracle-program-master/programs/oracle/src/state/product.rs
e94f8f9cc9a2ffb00e46ca58a5e91c810e9d220386886078fb83e1d5d3485a78	oracle-program-master/programs/oracle/src/errors.rs

1.3. Audit methodology

Our security audit process for RENEC smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using our in-house smart contract security analysis tool.
- Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that were considered during the audit of the RENEC smart contract:

- Arithmetic Overflow and Underflow
- Signer checks
- Ownership checks
- Rent exemption checks
- Account confusions
- Bump seed canonicalization
- Closing account
- Signed invocation of unverified programs
- Numerical precision errors
- Logic Flaws

For vulnerabilities, we categorize the findings into categories as listed in table below, depending on their severity level:

SEVERITY LEVEL	DESCRIPTION
CRITICAL	A vulnerability that can disrupt the contract functioning; creates a critical risk to the contract; required to be fixed immediately.
HIGH	A vulnerability that could affect the desired outcome of executing the contract with high impact; needs to be fixed with high priority.
MEDIUM	A vulnerability that could affect the desired outcome of executing the contract with medium impact in a specific scenario; needs to be fixed.
LOW	An issue that does not have a significant impact, can be considered as less important.

Table 1. Severity levels

1.4. Disclaimer

Please note that security auditing cannot uncover all existing vulnerabilities, and even an audit in which no vulnerabilities are found is not a guarantee for a 100% secure smart contract. However, auditing allows discovering vulnerabilities that were unobserved, overlooked during development and areas where additional security measures are necessary.

2. AUDIT RESULT

2.1. Overview

The Remitano Oracle and Pegged-Fiat programs was written in `Rust` programming language and `Anchor` framework.

There are two main programs in the audit scope. They are `Oracle` and `Pegged-Fiat`

2.1.1. Oracle program

This program provides price for a product (a pair of tokens) by using many oracle sources. Each pair is assigned to some publishers (oracle relayers) which will constantly fetch prices from many oracle services and update the price to the pair. Each product has a window time in which, the price will be calculated by average price of all prices updated by publishers.

2.1.2. Pegged-Fiat program

This program allows users to lock their fiat-based tokens to mint other tokens based on prices provided by `Oracle` program. After admin initializes and adds pairs, users can lock their base tokens (e.g., reUSD) to mint quote tokens (e.g., reVND) based on the price from `Oracle` program.

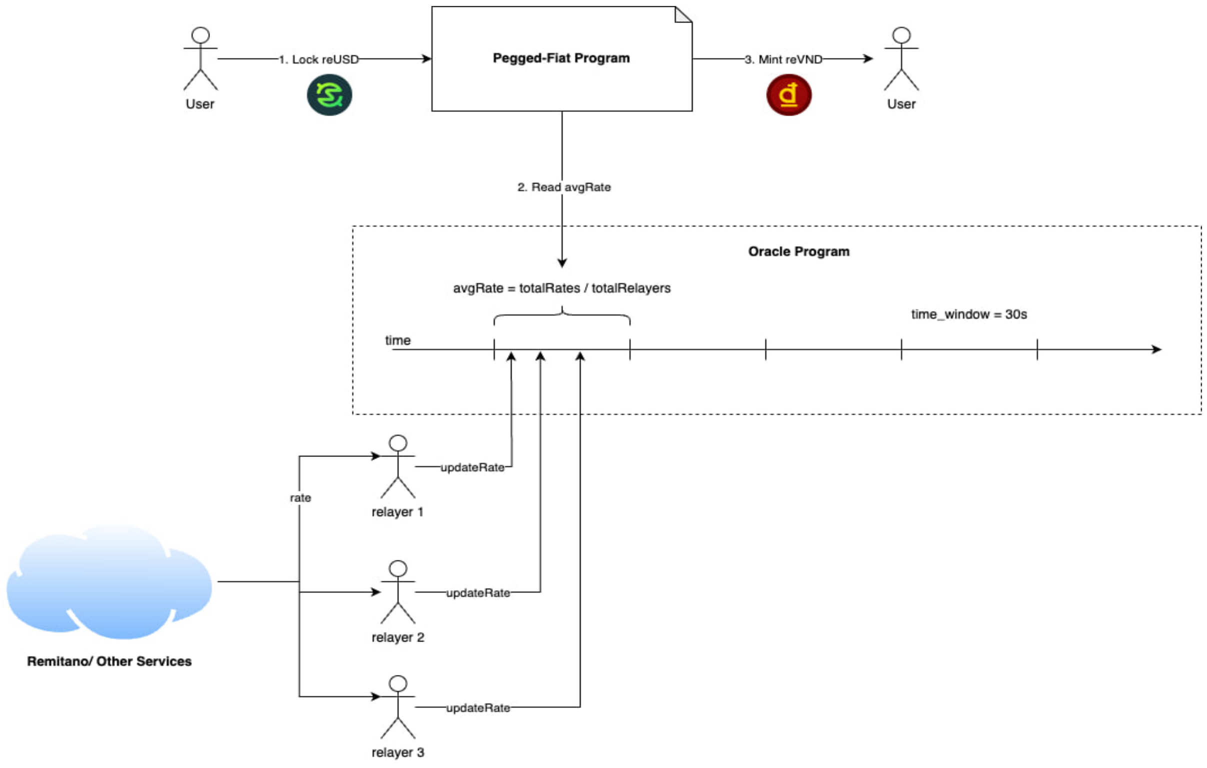

Image 1. Remitano Oracle and Pegged-Fiat programs graph

2.2. Findings

During the audit process, the audit team had identified some vulnerable issues in the given version of Remitano Oracle and Pegged-Fiat programs.

Remitano fixed the code, according to Verichains's draft report.

2.2.1. Unable to use `product` account after change base/quote HIGH

Affected files:

- set_token_mints.rs

The `product` account is a PDA account derived from `PRODUCT_SEED`, `controller`, `product.version`, `product.quote_mint`, `product.base_mint`. When `set_product_tokens` to another tokens for base/quote, the `product` account will be unable to pass the check of `Anchor` because the seeds have been changed.

```
pub fn set_product_tokens(ctx: Context<SetTokenMints>) -> ProgramResult {
    instructions::set_token_mints::handler(ctx)
}

#[account(mut,
    seeds = [
        PRODUCT_SEED,
        controller.key().as_ref(),
        product.version.to_le_bytes().as_ref(),
        product.quote_mint.as_ref(),
        product.base_mint.as_ref(),
    ],
    bump = product.bump[0],
    has_one = controller
)]
pub product: Box<Account<'info, Product>>

pub fn handler(ctx: Context<SetTokenMints>) -> ProgramResult {
    let product = &mut ctx.accounts.product;
    let quote_mint = &ctx.accounts.quote_mint;
    let base_mint = &ctx.accounts.base_mint;

    product.set_token_mints(
        quote_mint.key(),
        base_mint.key(),
    )?;

    Ok(())
}

pub fn set_token_mints(
    &mut self,
    quote_mint: Pubkey,
    base_mint: Pubkey,
) -> ProgramResult {
    self.quote_mint = quote_mint;
    self.base_mint = base_mint;

    Ok(())
}
```

RECOMMENDATION

Add new products for others base/quote.

UPDATES

- *Jul 27, 2023*: This issue has been acknowledged and fixed.

2.2.2. A single malicious/faulty publisher can manipulate the price HIGH

Affected files:

- post_price.rs

In current implementation, a publisher can update price many times or update price without calculating average when updating in a new window. The 5% different (specifies in the document) from updating price and average price has not been implemented yet. So the price will be unreliable because a single malicious/faulty publisher can manipulate price by updating price to lower price many times and defeat the purpose of having average price from different sources.

In other oracle systems, the price is only updated when an enough quorum of relayers' answer meets and each relayer can only post the answer one time.

```
pub fn handler(ctx: Context<PostPrice>, new_price: u64) -> ProgramResult {
    let price = &mut ctx.accounts.price;
    let publisher = &mut ctx.accounts.publisher;
    let product = &ctx.accounts.product;

    require!(
        new_price >= product.min_price && new_price <= product.max_price,
        ErrorCode::InputInvalidPrice
    );

    require!(product.status == ProductStatus::Online, ErrorCode::UnavailableProduct);

    let clock = Clock::get()?;
    let now = to_timestamp_u64(clock.unix_timestamp)?;
    let now_window_time = now
        .checked_div(product.window_size)
        .ok_or(ErrorCode::Overflow)?;

    let last_window_time = price
        .timestamp
        .checked_div(product.window_size)
        .ok_or(ErrorCode::Overflow)?;

    price.update_price(new_price, last_window_time != now_window_time, now)?;
    publisher.update_last_push_timestamp(now)?;

    Ok(())
}

pub fn update_price(&mut self, new_price: u64, is_refresh: bool, now: u64) -> ProgramResult
{
    if is_refresh {
        self.prev_price = self.price;
        self.prev_timestamp = self.timestamp;

        self.timestamp = now;
        self.num_publishers = 1;
        self.price = new_price;
    } else {
        self.timestamp = now;

        let mut sum_prices = self
            .price
            .checked_mul(u64::from(self.num_publishers))
            .ok_or(ErrorCode::Overflow)?;

        sum_prices = sum_prices
            .checked_add(new_price)
            .ok_or(ErrorCode::Overflow)?;

        self.num_publishers = self
            .num_publishers
            .checked_add(1)
            .ok_or(ErrorCode::Overflow)?;

        self.price = sum_prices
            .checked_div(u64::from(self.num_publishers))
            .ok_or(ErrorCode::Overflow)?;
    }

    Ok(())
}
```

RECOMMENDATION

Add 5% different price specifies in the document. Limit each publisher can post the price 1 time and update the price only when an enough quorum of relayers' answer meets.

UPDATES

- *Jul 27, 2023*: Remitano updated the code, added the check that the update price must not be greater than 5% of average price within 60 seconds window time and each publisher can only update the price 1 time in a window.

2.2.3. Wrong `EXPO` condition MEDIUM

Affected files:

- add_product.rs

There is a wrong condition when adding product `MIN_EXPO >= -10`, it must be `expo >= MIN_EXPO`.

```
pub fn handler(
    ctx: Context<AddProduct>,
    product_bump: u8,
    price_bump: u8,
    quote_currency: String,
    base_currency: String,
    asset_type: AssetType,
    expo: i32,
    max_price: u64,
    min_price: u64,
    window_size: u64,
) -> ProgramResult {
    require!(
        min_price > 0 && max_price >= min_price,
        ErrorCode::InputInvalidPrice
    );
    require!(
        MIN_EXPO >= -10 && expo <= MAX_EXPO, // Wrong condition
        ErrorCode::InputInvalidPrice
    );
    ...
}
```

RECOMMENDATION

Fix the condition to `expo >= MIN_EXPO`.

```
pub fn handler(
    ctx: Context<AddProduct>,
    product_bump: u8,
    price_bump: u8,
    quote_currency: String,
    base_currency: String,
    asset_type: AssetType,
    expo: i32,
    max_price: u64,
    min_price: u64,
    window_size: u64,
) -> ProgramResult {
    require!(
        min_price > 0 && max_price >= min_price,
        ErrorCode::InputInvalidPrice
    );
    require!(
        expo >= MIN_EXPO && expo <= MAX_EXPO, // Fix here
        ErrorCode::InputInvalidPrice
    );
    ...
}
```

UPDATES

- *Jul 27, 2023*: This issue has been acknowledged and fixed.

2.2.4. Anyone can initialize the program INFORMATIVE

Affected files:

- initialize.rs

In current implementation, anyone can initialize the program to add new `controller` as long as it is a different `version`. If this is not intended, let the admin initializes the program once then create different versions later.

```
#[derive(Accounts)]
#[instruction(bump: u8,  version: u16)]
pub struct Initialize<'info>{
    #[account(mut)]
    pub authority: Signer<'info>,

    #[account(init,
        seeds = [
            ORACLE_SEED,
            version.to_le_bytes().as_ref(),
        ],
        bump,
        payer = authority,
        space = Controller::LEN)]
    pub controller: Box<Account<'info, Controller>>,
    pub system_program: Program<'info, System>,
    pub rent: Sysvar<'info, Rent>,
}
```

UPDATES

- *Jul 27, 2023*: This issue has been acknowledged and fixed. The version has been disabled for now.

2.2.5. Should check status of product and price INFORMATIVE

Affected files:

- lock_n_mint.rs

When get the oracle price, the program should check `status` of both `product` and `price` to make sure that the product and price of oracle program is valid.

```
fn get_oracle_price<'info>(
    oracle_product: &AccountInfo<'info>,
    oracle_price: &AccountInfo<'info>,
) -> Result<PriceCalculator, ProgramError> {
    let mut oracle_product_data: &[u8] = &oracle_product.try_borrow_data()?;
    let oracle_product_info = oracle_states::Product::try_deserialize(&mut 
oracle_product_data)?;

    require!(
        oracle_product_info.price_account.eq(&oracle_price.key()),
        ErrorCode::InvalidPriceOfProductOracle
    );

    // check product status
    require!(oracle_product_info.status == oracle_states::ProductStatus::Online, 
ErrorCode::UnavailableProduct);

    let mut oracle_price_data: &[u8] = &oracle_price.try_borrow_data()?;
    let oracle_price_info = oracle_states::Price::try_deserialize(&mut oracle_price_data)?;

    // check price status
    require!(oracle_price_info.status == oracle_states::PriceStatus::Online, 
ErrorCode::UnavailablePrice);

    let clock = Clock::get()?;
    let _now = to_timestamp_u64(clock.unix_timestamp)?;

    // not elaped over 60s
    // require!(now - STALE_AFTER_SECS_ELAPSED <= oracle_price_info.timestamp, 
ErrorCode::PriceTooOld);
```

```
    let price_calculator = PriceCalculator::new(oracle_price_info.price,
oracle_product_info.expo)?;
    Ok(price_calculator)
}
```

UPDATES

- *Jul 27, 2023*: This issue has been acknowledged and fixed.

2.2.6. Wrong comments INFORMATIVE

Affected files:

- lock_n_mint.rs

There are some wrong comments in `lock_n_mint` handler, the correct must be `lock base` and `mint quote`.

```
pub fn handler(
    ctx: Context<LockNMint>,
    lock_amount: u64,
    user_profile_bump: u8,
    lock_profile_bump: u8,
) -> ProgramResult {
    ...
    // lock quote
    transfer_from_owner_to_vault(
        user,
        &ctx.accounts.lock_token_user,
        &ctx.accounts.lock_token_vault,
        token_program,
        lock_amount,
    )?;

    // mint base
    pair_mint_token_to_account(
        pair,
        mint_token_user,
        mint_token,
        token_program,
        mint_amount,
    )?;
    ...
}
```

UPDATES

- *Jul 31, 2023*: This issue has been acknowledged.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Jul 31, 2023*	Public Report	Verichains Lab

Table 2. Report versions history