BLOCKSEC

Security Audit
Report for Ref Contract and Ref Dcl

Date: July 30, 2024 **Version:** 2.0

Contact: contact@blocksec.com

Contents

Report Manifest

Item	Description
Client	Ref Finance
Target	Ref Contract and Ref Dcl

Version History

Version	Date	Description
1.0	July 5, 2024	First release
2.0	July 30, 2024	Second release

Signature

Digitally signed by: BlockSec (https://blocksec.com)

SHA1 digest of public key: BAB106D3708BC6DE0963CE05E9478C24B326AE1C

Date: 2024-07-30 06:48:09+0000

About BlockSec BlockSec focuses on the security of the blockchain ecosystem and collaborates with leading DeFi projects to secure their products. BlockSec is founded by top-notch security researchers and experienced experts from both academia and industry. They have published multiple blockchain security papers in prestigious conferences, reported several zero-day attacks of DeFi applications, and successfully protected digital assets that are worth more than 14 million dollars by blocking multiple attacks. They can be reached at Email, Twitter and Medium.

Chapter 1 Introduction

1.1 About Target Contracts

Information	Description
Type	Smart Contract
Language	Rust
Approach	Semi-automatic and manual verification

The target of this audit is the code repository of Ref Contract [1] and Ref Dcl [2] of Ref Finance. Note that, we did **NOT** audit all the modules in the repository. The modules covered by this audit report include `ref-contracts/ref-contracts/ref-exchange/src` folder and `ref-dcl/contracts/dcl/src` contract only. Specifically, the files covered in this audit include:

```
ref-dcl/contracts/dcl/src/api/dcl_api.rs
ref-dcl/contracts/dcl/src/dcl/pool.rs
ref-dcl/contracts/dcl/src/dcl/swap.rs
ref-dcl/contracts/dcl/src/api/token_receiver.rs

ref-contracts/ref-exchange/src/degen_swap/degen.rs
ref-contracts/ref-exchange/src/degen_swap/math.rs
ref-contracts/ref-exchange/src/degen_swap/mod.rs
ref-contracts/ref-exchange/src/degen_swap/price_oracle.rs
ref-contracts/ref-exchange/src/degen_swap/pyth_oracle.rs
ref-contracts/ref-exchange/src/rated_swap/sfrax_rate.rs
ref-contracts/ref-exchange/src/pool_limit_info.rs
ref-contracts/ref-exchange/src/account_deposit.rs
ref-contracts/ref-exchange/src/action.rs
ref-contracts/ref-exchange/src/custom_keys.rs
ref-contracts/ref-exchange/src/errors.rs
ref-contracts/ref-exchange/src/lib.rs
ref-contracts/ref-exchange/src/oracle.rs
ref-contracts/ref-exchange/src/owner.rs
ref-contracts/ref-exchange/src/pool.rs
ref-contracts/ref-exchange/src/simple_pool.rs
ref-contracts/ref-exchange/src/token_receiver.rs
ref-contracts/ref-exchange/src/views.rs
```

Listing 1.1: Audit Scope for this Report

The auditing process is iterative. Specifically, we would audit the commits that fix the discovered issues. If there are new issues, we will continue this process. The commit SHA values during the audit are shown in the following table. Our audit report is responsible for the code in the initial version (`Version 1` and `Version 3`), as well as new code (in the following versions) to fix issues in the audit report.

[1] https://github.com/ref-finance/ref-contracts/tree/degen-pool

[2] https://github.com/ref-finance/ref-dcl/tree/open_create_pool

Project	Version	Commit Hash
Ref Contract	Version 1	37150859766902dc123db58066cc64305f259e42
	Version 2	5090a7ad4ec7d333f7c6d1bb0b7ccf3e929098a9
	Version 3	93baf5e4db633a346790c2ec4f62796b9e28f1ec
	Version 4	2f1577ec4dcf9b77e27c4373706bd46f64a235f4
Ref Dcl	Version 1	ac89456c21b825b92bbadc9ba18f82663f240f70
	Version 2	47267c695f8144b8cc0a9ed7dd7624b7d34cd56b

1.2 Disclaimer

This audit report does not constitute investment advice or a personal recommendation. It does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset. Any entity should not rely on this report in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset.

This audit report is not an endorsement of any particular project or team, and the report does not guarantee the security of any particular project. This audit does not give any warranties on discovering all security issues of the smart contracts, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues. As one audit cannot be considered comprehensive, we always recommend proceeding with independent audits and a public bug bounty program to ensure the security of smart contracts.

The scope of this audit is limited to the code mentioned in Section 1.1. Unless explicitly specified, the security of the language itself (e.g., the solidity language), the underlying compiling toolchain and the computing infrastructure are out of the scope.

1.3 Procedure of Auditing

We perform the audit according to the following procedure.

- **Vulnerability Detection** We first scan smart contracts with automatic code analyzers, and then manually verify (reject or confirm) the issues reported by them.
- **Semantic Analysis** We study the business logic of smart contracts and conduct further investigation on the possible vulnerabilities using an automatic fuzzing tool (developed by our research team). We also manually analyze possible attack scenarios with independent auditors to cross-check the result.
- **Recommendation** We provide some useful advice to developers from the perspective of good programming practice, including gas optimization, code style, and etc.

We show the main concrete checkpoints in the following.

1.3.1 Software Security

* Reentrancy
* DoS
* Access control
* Data handling and data flow

- Exception handling
- Untrusted external call and control flow
- Initialization consistency
- Events operation
- Error-prone randomness
- Improper use of the proxy system

1.3.2 DeFi Security

- Semantic consistency
- Functionality consistency
- Permission management
- Business logic
- Token operation
- Emergency mechanism
- Oracle security
- Whitelist and blacklist
- Economic impact
- Batch transfer

1.3.3 NFT Security

- Duplicated item
- Verification of the token receiver
- Off-chain metadata security

1.3.4 Additional Recommendation

- Gas optimization
- Code quality and style

Note *The previous checkpoints are the main ones. We may use more checkpoints during the auditing process according to the functionality of the project.*

1.4 Security Model

To evaluate the risk, we follow the standards or suggestions that are widely adopted by both industry and academy, including OWASP Risk Rating Methodology [3] and Common Weakness Enumeration [4]. The overall *severity* of the risk is determined by *likelihood* and *impact*. Specifically, likelihood is used to estimate how likely a particular vulnerability can be uncovered and exploited by an attacker, while impact is used to measure the consequences of a successful exploit.

[3] https://owasp.org/www-community/OWASP_Risk_Rating_Methodology

[4] https://cwe.mitre.org/

In this report, both likelihood and impact are categorized into two ratings, i.e., *high* and *low* respectively, and their combinations are shown in Table 1.1.

Table 1.1: Vulnerability Severity Classification

Impact		Likelihood: *High*	Likelihood: *Low*
	High	High	Medium
	Low	Medium	Low

Accordingly, the severity measured in this report are classified into three categories: **High**, **Medium**, **Low**. For the sake of completeness, **Undetermined** is also used to cover circumstances when the risk cannot be well determined.

Furthermore, the status of a discovered item will fall into one of the following four categories:

- **Undetermined** No response yet.
- **Acknowledged** The item has been received by the client, but not confirmed yet.
- **Confirmed** The item has been recognized by the client, but not fixed yet.
- **Fixed** The item has been confirmed and fixed by the client.

Chapter 2 Findings

In total, we found **four** potential security issues. Besides, we have **two** recommendations.

- High Risk: 3
- Low Risk: 1
- Recommendation: 2

ID	Severity	Description	Category	Status
1	High	Inaccurate output amount calculation in function `internal_swap_by_output()`	DeFi Security	Fixed
2	High	Lack of state update in function `internal_quote_by_output()`	DeFi Security	Fixed
3	High	Lack of check on pool TVL after adding liquidity	DeFi Security	Fixed
4	Low	Lack of TVL price valid check	DeFi Security	Fixed
5	-	Redundant check in function `swap()`	Recommendation	Fixed
6	-	Duplicated price requests	Recommendation	Fixed

The details are provided in the following sections.

2.1 DeFi Security

2.1.1 Inaccurate output amount calculation in function `internal_swap_by_output()`

Severity High

Status Fixed at `Version 2`

Introduced by `Version 1`

Description In the function `internal_swap_by_output()`, the `actual_output_amount` variable, which is calculated and updated during the swap process, is used to determine the amount of output tokens sent to the user. However, this variable does not accurately represent the actual output token amount during the swap.

```
238   pub fn internal_swap_by_output(
239       &mut self,
240       account_id: &AccountId,
241       pool_ids: Vec<PoolId>,
242       input_token: &AccountId,
243       max_input_amount: Balance,
244       output_token: &AccountId,
245       output_amount: Balance,
246       skip_unwrap_near: Option<bool>,
247       client_echo: Option<String>,
248   ) -> Balance {
249       pool_ids.iter().for_each(|pool_id| {
250           self.assert_pool_running(&self.internal_unwrap_pool(pool_id));
251           let (token_x, token_y, _) = pool_id.parse_pool_id();
252           self.assert_no_frozen_tokens(&[token_x, token_y]);
```

```
253         });
254
255         let protocol_fee_rate = self.data().protocol_fee_rate;
256         let vip_info = self.data().vip_users.get(account_id);
257         let (actual_input_token, actual_input_amount, actual_output_amount) = {
258             let mut next_desire_token = output_token.clone();
259             let mut next_desire_amount = output_amount;
260             let mut actual_output_amount = output_amount;
261             for pool_id in pool_ids.iter() {
262                 let mut pool = self.internal_unwrap_pool(&pool_id);
263
264                 let pool_fee = pool.get_pool_fee_by_user(&vip_info);
265
266                 if next_desire_token.eq(&pool.token_x) {
267                     let (need_amount, acquire_amount, is_finished, total_fee, protocol_fee) = pool.
                            internal_y_swap_x_desire_x(pool_fee, protocol_fee_rate, next_desire_amount,
                            800001, false);
268                     if !is_finished {
269                         env::panic_str(&format!("ERR_TOKEN_{}_NOT_ENOUGH", pool.token_x.to_string().
                                to_uppercase()));
270                     }
271
272                     pool.total_y += need_amount;
273                     pool.total_x -= acquire_amount;
274                     pool.volume_y_in += U256::from(need_amount);
275                     pool.volume_x_out += U256::from(acquire_amount);
276
277                     actual_output_amount = acquire_amount;
278                     next_desire_token = pool.token_y.clone();
279                     next_desire_amount = need_amount;
280
281                     Event::SwapDesire {
282                         swapper: account_id,
283                         token_in: &pool.token_y,
284                         token_out: &pool.token_x,
285                         amount_in: &U128(need_amount),
286                         amount_out: &U128(acquire_amount),
287                         pool_id: &pool.pool_id,
288                         total_fee: &U128(total_fee),
289                         protocol_fee: &U128(protocol_fee),
290                     }
291                     .emit();
292                 } else if next_desire_token.eq(&pool.token_y) {
293                     let (need_amount, acquire_amount, is_finished, total_fee, protocol_fee) = pool.
                            internal_x_swap_y_desire_y(pool_fee, protocol_fee_rate, next_desire_amount,
                            -800001, false);
294                     if !is_finished {
295                         env::panic_str(&format!("ERR_TOKEN_{}_NOT_ENOUGH", pool.token_y.to_string().
                                to_uppercase()));
296                     }
297
298                     pool.total_x += need_amount;
299                     pool.total_y -= acquire_amount;
```

```
300                 pool.volume_x_in += U256::from(need_amount);
301                 pool.volume_y_out += U256::from(acquire_amount);
302
303                 actual_output_amount = acquire_amount;
304                 next_desire_token = pool.token_x.clone();
305                 next_desire_amount = need_amount;
306
307                 Event::SwapDesire {
308                     swapper: account_id,
309                     token_in: &pool.token_x,
310                     token_out: &pool.token_y,
311                     amount_in: &U128(need_amount),
312                     amount_out: &U128(acquire_amount),
313                     pool_id: &pool.pool_id,
314                     total_fee: &U128(total_fee),
315                     protocol_fee: &U128(protocol_fee),
316                 }
317                 .emit();
318             } else {
319                 env::panic_str(E404_INVALID_POOL_IDS);
320             }
321             self.internal_set_pool(&pool_id, pool);
322         }
323         (next_desire_token, next_desire_amount, actual_output_amount)
324     };
325     require!(input_token == &actual_input_token, E213_INVALID_INPUT_TOKEN);
326     require!(actual_input_amount <= max_input_amount, E204_SLIPPAGE_ERR);
327
328     if actual_output_amount > 0 {
329         if let Some(msg) = client_echo {
330             self.process_ft_transfer_call(account_id, &output_token, actual_output_amount, msg)
                    ;
331         } else {
332             self.process_transfer(account_id, &output_token, actual_output_amount,
                    skip_unwrap_near);
333         }
334     }
335
336     actual_input_amount
337 }
```

Listing 2.1: ref-dcl/contracts/dcl/src/dcl/swap.rs

Impact The inaccurate output amount calculation in function `internal_swap_by_output()` leads to incorrect internal accounting. This allows attackers to receive more tokens than they should be entitled to.

Suggestion Revise the output token amount accordingly.

2.1.2 Lack of state update in function `internal_quote_by_output()`

Severity High

Status Fixed in `Version 2`

Introduced by `Version 1`

Description In the `internal_quote_by_output()` function, the state of the `pool` modified during the quoting process is not written back to the `pool_cache`. Therefore, if the same pool is accessed again, the retrieved state will be incorrect.

```
71  pub fn internal_quote_by_output(
72      &self,
73      pool_cache: &mut HashMap<PoolId, Pool>,
74      vip_info: Option<HashMap<PoolId, u32>>,
75      pool_ids: Vec<PoolId>,
76      input_token: AccountId,
77      output_token: AccountId,
78      output_amount: U128,
79      tag: Option<String>,
80  ) -> QuoteResult {
81      let quote_failed = QuoteResult {
82          amount: 0.into(),
83          tag: tag.clone(),
84      };
85      if self.data().state == RunningState::Paused {
86          return quote_failed;
87      }
88
89      let protocol_fee_rate = self.data().protocol_fee_rate;
90
91      let (actual_input_token, actual_input_amount) = {
92          let mut next_desire_token = output_token;
93          let mut next_desire_amount = output_amount.0;
94          for pool_id in pool_ids {
95              let mut pool = pool_cache.remove(&pool_id).unwrap_or(self.internal_unwrap_pool(&
                    pool_id));
96              if pool.state == RunningState::Paused ||
97                  self.data().frozenlist.contains(&pool.token_x) || self.data().frozenlist.
                        contains(&pool.token_y) {
98                  return quote_failed;
99              }
100
101             let pool_fee = pool.get_pool_fee_by_user(&vip_info);
102
103             let is_finished = if next_desire_token.eq(&pool.token_x) {
104                 let (need_amount, _, is_finished, _, _) = pool.internal_y_swap_x_desire_x(
                        pool_fee, protocol_fee_rate, next_desire_amount, 800001, true);
105                 next_desire_token = pool.token_y.clone();
106                 next_desire_amount = need_amount;
107                 is_finished
108             } else if next_desire_token.eq(&pool.token_y) {
109                 let (need_amount, _, is_finished, _, _) = pool.internal_x_swap_y_desire_y(
                        pool_fee, protocol_fee_rate, next_desire_amount, -800001, true);
110                 next_desire_token = pool.token_x.clone();
111                 next_desire_amount = need_amount;
112                 is_finished
```

```
113                 } else {
114                     return quote_failed;
115                 };
116                 if !is_finished {
117                     return quote_failed;
118                 }
119             }
120             (next_desire_token, next_desire_amount)
121         };
122         if input_token != actual_input_token {
123             return quote_failed;
124         }
125         QuoteResult {
126             amount: actual_input_amount.into(),
127             tag,
128         }
129     }
```

Listing 2.2: ref-dcl/contracts/dcl/src/dcl/swap.rs

Impact This can lead to erroneous results if duplicate pool ids are provided.

Suggestion Write back the updated `pool` state to `pool_cache`.

2.1.3 Lack of check on pool TVL after adding liquidity

Severity High

Status Fixed in `Version 4`

Introduced by `Version 3`

Description In the `ft_on_transfer()` function, when handling the `HotZap` message, liquidity may be added to the Degen Pool. However, there is no check to ensure that the pool's TVL does not exceed the specified limit after the liquidity is added.

```
136     TokenReceiverMessage::HotZap {
137         referral_id,
138         hot_zap_actions,
139         add_liquidity_infos
140     } => {
141         assert!(hot_zap_actions.len() > 0 && add_liquidity_infos.len() > 0);
142         let sender_id: AccountId = sender_id.into();
143         let mut account = self.internal_unwrap_account(&sender_id);
144         let referral_id = referral_id.map(|x| x.to_string());
145         let out_amounts = self.internal_direct_actions(
146             token_in,
147             amount.0,
148             referral_id,
149             &hot_zap_actions,
150         );
151
152         let mut token_cache = TokenCache::new();
153         for (out_token_id, out_amount) in out_amounts {
154             token_cache.add(&out_token_id, out_amount);
```

```
155         }
156
157         let prev_storage = env::storage_usage();
158         for add_liquidity_info in add_liquidity_infos {
159             let mut pool = self.pools.get(add_liquidity_info.pool_id).expect(ERR85_NO_POOL);
160             let tokens_in_pool = match &pool {
161                 Pool::SimplePool(p) => p.token_account_ids.clone(),
162                 Pool::RatedSwapPool(p) => p.token_account_ids.clone(),
163                 Pool::StableSwapPool(p) => p.token_account_ids.clone(),
164                 Pool::DegenSwapPool(p) => p.token_account_ids.clone(),
165             };
166
167             let mut add_liquidity_amounts = add_liquidity_info.amounts.iter().map(|v| v.0).collect
                    ();
168
169             match pool {
170                 Pool::SimplePool(_) => {
171                     pool.add_liquidity(
172                         &sender_id,
173                         &mut add_liquidity_amounts,
174                         false
175                     );
176                     let min_amounts = add_liquidity_info.min_amounts.expect("Need input min_amounts"
                            );
177                     // Check that all amounts are above request min amounts in case of front running
                            that changes the exchange rate.
178                     for (amount, min_amount) in add_liquidity_amounts.iter().zip(min_amounts.iter())
                            {
179                         assert!(amount >= &min_amount.0, "{}", ERR86_MIN_AMOUNT);
180                     }
181                 },
182                 Pool::StableSwapPool(_) | Pool::RatedSwapPool(_) | Pool::DegenSwapPool(_) => {
183                     let min_shares = add_liquidity_info.min_shares.expect("Need input min_shares");
184                     pool.add_stable_liquidity(
185                         &sender_id,
186                         &add_liquidity_amounts,
187                         min_shares.into(),
188                         AdminFees::new(self.admin_fee_bps),
189                         false
190                     );
191                 }
192             };
193
194             for (cost_token_id, cost_amount) in tokens_in_pool.iter().zip(add_liquidity_amounts.
                    into_iter()) {
195                 token_cache.sub(cost_token_id, cost_amount);
196             }
197
198             self.pools.replace(add_liquidity_info.pool_id, &pool);
199         }
200
201         if env::storage_usage() > prev_storage {
202             let storage_cost = (env::storage_usage() - prev_storage) as Balance * env::
```

```
                storage_byte_cost();
203         account.near_amount = account.near_amount.checked_sub(storage_cost).expect(
                ERR11_INSUFFICIENT_STORAGE);
204     }
205
206     for (remain_token_id, remain_amount) in token_cache.0.iter() {
207         account.deposit(remain_token_id, *remain_amount);
208     }
209
210     self.internal_save_account(&sender_id, account);
211
212     env::log(
213         format!(
214             "HotZap remain internal account assets: {:?}",
215             token_cache.0
216         )
217         .as_bytes(),
218     );
219
220     PromiseOrValue::Value(U128(0))
221 }
```

Listing 2.3: ref-contract/ref-exchange/src/token_receiver.rs

Impact Degen Pool's TVL may exceed the limit.

Suggestion Add check accordingly.

2.1.4 Lack of TVL price valid check

Severity Low

Status Fixed in `Version 4`

Introduced by `Version 3`

Description The view function `get_degen_pool_tvl()` calculates the Degen Pool's TVL using the prices of the tokens. However, it doesn't check if the price is valid.

```
848 pub fn get_degen_pool_tvl(&self, pool_id: u64) -> U128 {
849     self.pools.get(pool_id).expect(ERR85_NO_POOL).get_tvl().into()
850 }
```

Listing 2.4: ref-contract/ref-exchange/src/views.rs

Impact Users may get expired TVL.

Suggestion Add check accordingly.

2.2 Additional Recommendation

2.2.1 Redundant check in function `swap()`

Status Fixed in `Version 2`

Introduced by Version 1

Description The `assert_contract_running()` check is redundant in functions `swap()` and `swap_by_output()` since the same check will be performed in the function `execute_actions()`.

```
269   /// Execute set of swap actions between pools.
270   /// If referrer provided, pays referral_fee to it.
271   /// If no attached deposit, outgoing tokens used in swaps must be whitelisted.
272   #[payable]
273   pub fn swap(&mut self, actions: Vec<SwapAction>, referral_id: Option<ValidAccountId>) -> U128
          {
274       self.assert_contract_running();
275       U128(
276           self.execute_actions(
277               actions
278                   .into_iter()
279                   .map(|swap_action| Action::Swap(swap_action))
280                   .collect(),
281               referral_id,
282           )
283           .to_amount(),
284       )
285   }
286
287   /// Execute set of swap_by_output actions between pools.
288   /// If referrer provided, pays referral_fee to it.
289   /// If no attached deposit, outgoing tokens used in swaps must be whitelisted.
290   #[payable]
291   pub fn swap_by_output(&mut self, actions: Vec<SwapByOutputAction>, referral_id: Option<
          ValidAccountId>) -> U128 {
292       self.assert_contract_running();
293       U128(
294           self.execute_actions(
295               actions
296                   .into_iter()
297                   .map(|swap_by_output_action| Action::SwapByOutput(swap_by_output_action))
298                   .collect(),
299               referral_id,
300           )
301           .to_amount(),
302       )
303   }
```

Listing 2.5: ref-contracts/ref-exchange/src/lib.rs

Suggestion Remove the redundant check.

2.2.2 Duplicated price requests

Status Fixed in Version 2

Introduced by Version 1

Description Every time the function `swap()` of `DegenSwapPool` is invoked, it requests price synchronization for all tokens in the pool from the oracles. However, since the `swap()` function

can be invoked multiple times within a single transaction, this may result in redundant token price requests, leading to unnecessary gas consumption.

```
561   pub fn swap(
562       &mut self,
563       token_in: &AccountId,
564       amount_in: Balance,
565       token_out: &AccountId,
566       min_amount_out: Balance,
567       fees: &AdminFees,
568       is_view: bool
569   ) -> Balance {
570
571       assert_ne!(token_in, token_out, "{}", ERR71_SWAP_DUP_TOKENS);
572       let in_idx = self.token_index(token_in);
573       let out_idx = self.token_index(token_out);
574       let result = self.internal_get_return(in_idx, amount_in, out_idx, &fees);
575       let amount_swapped = self.c_amount_to_amount(result.amount_swapped, out_idx);
576       assert!(
577           amount_swapped >= min_amount_out,
578           "{}",
579           ERR68_SLIPPAGE
580       );
581       if !is_view {
582           env::log(
583               format!(
584                   "Swapped {} {} for {} {}, total fee {}, admin fee {}",
585                   amount_in, token_in, amount_swapped, token_out,
586                   self.c_amount_to_amount(result.fee, out_idx),
587                   self.c_amount_to_amount(result.admin_fee, out_idx)
588               )
589               .as_bytes(),
590           );
591       }
592
593       self.c_amounts[in_idx] = result.new_source_amount;
594       self.c_amounts[out_idx] = result.new_destination_amount;
595       self.assert_min_reserve(self.c_amounts[out_idx]);
596
597       // Keeping track of volume per each input traded separately.
598       self.volumes[in_idx].input.0 += amount_in;
599       self.volumes[out_idx].output.0 += amount_swapped;
600
601       // handle admin fee.
602       if fees.admin_fee_bps > 0 && result.admin_fee > 0 {
603           let (exchange_share, referral_share) = if let Some((referral_id, referral_fee)) = &fees
                .referral_info {
604               if self.shares.contains_key(referral_id)
605               {
606                   self.distribute_admin_fee(&fees.exchange_id, referral_id, *referral_fee, out_idx
                        , result.admin_fee, is_view)
607               } else {
608                   self.distribute_admin_fee(&fees.exchange_id, referral_id, 0, out_idx, result.
```

```
                        admin_fee, is_view)
609             }
610         } else {
611             self.distribute_admin_fee(&fees.exchange_id, &fees.exchange_id, 0, out_idx, result.
                    admin_fee, is_view)
612         };
613         if !is_view {
614             if referral_share > 0 {
615                 env::log(
616                     format!(
617                         "Exchange {} got {} shares, Referral {} got {} shares",
618                         &fees.exchange_id, exchange_share, &fees.referral_info.as_ref().unwrap()
                                .0, referral_share,
619                     )
620                     .as_bytes(),
621                 );
622             } else {
623                 env::log(
624                     format!(
625                         "Exchange {} got {} shares, No referral fee",
626                         &fees.exchange_id, exchange_share,
627                     )
628                     .as_bytes(),
629                 );
630             }
631         }
632     }
633
634     if !is_view {
635         for token_id in self.token_account_ids.iter() {
636             let degen = global_get_degen(token_id);
637             degen.sync_token_price(token_id);
638         }
639     }
640
641     amount_swapped
642 }
```

Listing 2.6: ref-contracts/ref-exchange/src/lib.rs

Suggestion Add checks to verify if the token is currently in the midst of syncing its price, and only request an update from the oracle if it is not already in the sync process.


BOOST WEB3 THROUGH NEXT-GENERATION SECURITY & USABILITY INNOVATIONS