Security Assessment Report

Scope

December 16, 2024

Summary

The Sec3 team (formerly Soteria) was engaged to conduct a thorough security analysis of the Scope smart contracts.

The artifact of the audit was the source code of the following programs, excluding tests, in https://github.com/Kamino-Finance/scope/tree/3dba648.

The initial audit focused on the following versions and revealed 4 issues or questions.

program	type	commit
scope	Solana	`3dba648851fec203b1853cf765a388263ee22ff4`

This report provides a detailed description of the findings and their respective resolutions.

Table of Contents

Result Overview

Issue	Impact	Status
SCOPE		
[M-01] Incorrect timestamp retrieval in "switchboard_on_demand::get_price"	Medium	Resolved
[L-01] Inconsistent behavior in "reset_twap"	Low	Resolved
[I-01] Outdated AggregatorAccountData handling	Info	Acknowledged
[Q-01] Reference price design question	Question	Resolved

Findings in Detail

SCOPE

[M-01] Incorrect timestamp retrieval in "switchboard_on_demand::get_price"

In the current implementation of "switchboard_on_demand::get_price", the data of a Switchboard feed account of type "PullFeedAccountData" is deserialized, and the "result.timestamp" field is recorded as the price data's timestamp.

```
/* programs/scope/src/oracles/switchboard_on_demand.rs */
052 | let unix_timestamp = feed.result.timestamp.try_into().unwrap();
053 |
054 | Ok(DatedPrice {
055 |     price,
056 |     last_updated_slot,
057 |     unix_timestamp,
058 |     ..Default::default()
059 | })
```

However, the actual layout of the "CurrentResult" struct used in the "switchboard-on-demand" program differs from the layout assumed in the code.

Specifically, the location in the data where the code attempts to read the "timestamp" actually contains the fields "u8 num_samples", "u8 submission_idx", and 6 bytes of padding. This discrepancy results in incorrect retrieval of the timestamp.

Resolution

Fixed by commit "f85ec25".

SCOPE

[L-01] Inconsistent behavior in "reset_twap"

The "reset_twap" instruction, intended for admin use to reset the TWAP of a specific token, exhibits two inconsistencies in its implementation compared to other TWAP price update mechanisms:

1. **Incorrect Update of Metadata**

Instead of using the relevant information from "DatedPrice", the "reset_twap" instruction updates "last_update_slot" and "last_update_unix_timestamp" using the current slot and Unix timestamp. This divergence may introduce errors in the calculation of the smoothing factor, potentially leading to inaccurate EMA price computations.

```
/* programs/scope/src/handlers/handler_reset_twap.rs */
036 | crate::oracles::twap::reset_twap(
037 |     &mut oracle_twaps,
038 |     token,
039 |     price,
040 |     clock.unix_timestamp as u64,
041 |     clock.slot,
042 | )?;
```

2. **Inconsistent Handling of Data Points in update_ema_twap**

- The "update_ema_twap" function determines the absence of data points by checking whether "last_update_slot" equals 0.
- However, the "reset_twap" implementation does not reset "last_update_slot" to 0 while clearing the "updates_tracker_1h".
- As a result, the actual number of data points involved in calculations may exceed the count recorded in the tracker by one.

```
/* programs/scope/src/oracles/twap.rs */
124 | pub(super) fn update_ema_twap(
125 |     twap: &mut EmaTwap,
126 |     price: Price,
127 |     price_ts: u64,
128 |     price_slot: u64,
129 | ) -> ScopeResult<()> {
```

```
130 |     // Skip update if the price is the same as the last one
131 |     if price_slot > twap.last_update_slot {
132 |         if twap.last_update_slot == 0 {
133 |             twap.current_ema_1h = Decimal::from(price).to_scaled_val().unwrap();
134 |         } else {
162 |
163 | pub(super) fn reset_ema_twap(twap: &mut EmaTwap, price: Price, price_ts: u64, price_slot: u64) {
164 |     twap.current_ema_1h = Decimal::from(price).to_scaled_val().unwrap();
165 |     twap.last_update_slot = price_slot;
166 |     twap.last_update_unix_timestamp = price_ts;
167 |     twap.updates_tracker_1h = 0;
168 | }
```

Resolution

Fixed by commit "`1c38d2f`" in a private repo.

SCOPE

[I-01] Outdated AggregatorAccountData handling

In the "switchboard_v2::get_price", an older version of the "AggregatorAccountData" structure is utilized, which does not account for the newly added "resolution_mode" field.

When the "resolution_mode" field is set to "AggregatorResolutionMode::ModeSlidingResolution", subsequent checks for "min_oracle_results" and "latest_confirmed_round.num_success" can be omitted.

```
/*
https://github.com/switchboard-xyz/solana-sdk/blob/9e40fab/rust/switchboard-v2/src/aggregator.rs#L220-L230
*/
220 | pub fn get_result(&self) -> anchor_lang::Result<SwitchboardDecimal> {
221 |     if self.resolution_mode == AggregatorResolutionMode::ModeSlidingResolution {
222 |         return Ok(self.latest_confirmed_round.result);
223 |     }
224 |     let min_oracle_results = self.min_oracle_results;
225 |     let latest_confirmed_round_num_success = self.latest_confirmed_round.num_success;
226 |     if min_oracle_results > latest_confirmed_round_num_success {
227 |         return Err(SwitchboardError::InvalidAggregatorRound.into());
228 |     }
229 |     Ok(self.latest_confirmed_round.result)
230 | }
```

However, the absence of corresponding handling for this scenario does not introduce any security risks.

Resolution

The team clarified that Switchboard v2 has been deprecated. They have ceased using it, and its support will be removed from Scope.

SCOPE

[Q-01] Reference price design question

In the "`refresh_price_list`" instruction, for tokens with a reference price, if the new price deviates from the reference price by more than a certain threshold (5%), the price will not be updated. However, the TWAP is updated before this check is performed.

This means that even if the current price significantly deviates from the reference price, it will still impact the TWAP.

We would like to confirm if this behavior is intentional by design.

```
/* programs/scope/src/handlers/handler_refresh_prices.rs */
116 | if oracle_mappings.is_twap_enabled(token_idx) {
117 |     let _ = crate::oracles::twap::update_twap(&mut oracle_twaps, token_idx, &price)
118 |         .map_err(|_| msg!("Twap not found for token {}", token_idx));
119 | };
120 |
121 | // Only temporary load as mut to allow prices to be computed based on a scope chain
122 | // from the price feed that is currently updated
123 |
124 | let mut oracle_prices = ctx.accounts.oracle_prices.load_mut()?;
125 |
126 | // check that the price is close enough to the ref price is there is a ref price
127 | if oracle_mappings.ref_price[token_idx] != u16::MAX {
128 |     let ref_price =
129 |         oracle_prices.prices[usize::from(oracle_mappings.ref_price[token_idx])].price;
130 |     if let Err(diff_err) = check_ref_price_difference(price.price, ref_price) {
131 |         if fail_tx_on_error {
132 |             return Err(diff_err);
133 |         } else {
134 |             msg!(
135 |             "Price skipped as ref price check failed (token {token_idx}, type {price_type:?})",
136 |         );
137 |             continue;
138 |         }
139 |     }
140 | }
```

Resolution

The team clarified that this is the expected behavior. When the reference price is the TWAP itself, it ensures that the TWAP can still be updated, preventing a deadlock scenario where updates

would cease entirely.

Appendix: Methodology and Scope of Work

Assisted by the Sec3 Scanner developed in-house, the manual audit particularly focused on the following work items:

- Check common security issues.
- Check program logic implementation against available design specifications.
- Check poor coding practices and unsafe behavior.
- The soundness of the economics design and algorithm is out of scope of this work

DISCLAIMER

The instance report ("Report") was prepared pursuant to an agreement between Coderrect Inc. d/b/a Sec3 (the "Company") and the Client. This Report solely includes the results of a technical assessment of a specific build and/or version of the Client's code specified in the Report ("Assessed Code") by the Company. The sole purpose of the Report is to provide the Client with the results of the technical assessment of the Assessed Code. The Report does not apply to any other version and/or build of the Assessed Code. Regardless of the contents of the Report, the Report does not (and should not be interpreted to) provide any warranty, representation or covenant that the Assessed Code: (i) is error and/or bug free, (ii) has no security vulnerabilities, and/or (iii) does not infringe any third-party rights. Moreover, the Report is not, and should not be considered, an endorsement by the Company of the Assessed Code and/or of the Client. Finally, the Report should not be considered investment advice or a recommendation to invest in the Assessed Code and/or the Client.

This Report is considered null and void if the Report (or any portion thereof) is altered in any manner.

ABOUT

The Sec3 audit team comprises a group of computer science professors, researchers, and industry veterans with extensive experience in smart contract security, program analysis, testing, and formal verification. We are also building automated security tools that incorporate static analysis, penetration testing, and formal verification.

At Sec3, we identify and eliminate security vulnerabilities through the most rigorous process and aided by the most advanced analysis tools.

For more information, check out our website and follow us on twitter.