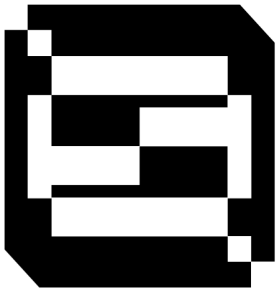

OFFSIDE Labs

Kamino Scope

Smart Contract Security Assessment

December 2023

Prepared for:

Hubble Protocol

Prepared by:

Offside Labs

Ronny Xing

Siji Feng

Contents

1 About Offside Labs

Offside Labs is a leading security research team, composed of top talented hackers from both academia and industry.

We possess a wide range of expertise in modern software systems, including, but not limited to, *browsers*, *operating systems*, *IoT devices*, and *hypervisors*. We are also at the forefront of innovative areas like *cryptocurrencies* and *blockchain technologies*. Among our notable accomplishments are remote jailbreaks of devices such as the **iPhone** and **PlayStation 4**, and addressing critical vulnerabilities in the **Tron Network**.

Our team actively engages with and contributes to the security community. Having won and also co-organized *DEFCON CTF*, the most famous CTF competition in the Web2 era, we also triumphed in the **Paradigm CTF 2023** within the Web3 space. In addition, our efforts in responsibly disclosing numerous vulnerabilities to leading tech companies, such as *Apple*, *Google*, and *Microsoft*, have protected digital assets valued at over **$300 million**.

In the transition towards Web3, Offside Labs has achieved remarkable success. We have earned over **$9 million** in bug bounties, and **three** of our innovative techniques were recognized among the **top 10 blockchain hacking techniques of 2022** by the Web3 security community.

https://offside.io/

https://github.com/offsidelabs

https://twitter.com/offside_labs

2 Executive Summary

Introduction

Offside Labs completed a security audit of *Kamino*'s *Scope* smart contracts, beginning on November 20, 2023, and concluding on December 8, 2023.

Scope Project Overview

Scope is a price oracle aggregator operating on the Solana network. It copies data from multiple on-chain oracles' accounts into one "price feed".

Scope pre-validates the prices with a preset of rules and performs the update only if they meet the criteria.

Audit Scope

The assessment scope primarily includes the smart contracts of the *Scope* program for the *Kamino* project by Nov 24, 2023.

The audit is based on the following specific branches and commit hashes of the codebase repositories:

- *Scope*
  - Branch: master
  - Commit Hash: 1e3877cdf22548bbda3a661af2af3d34dbce78c0
  - Codebase Link

We listed the files we have audited below:

- *Scope*
  - programs/scope/src/**/*.rs

Findings

The security audit revealed:

- 1 critical issue
- 3 medium issues
- 2 low issues
- 1 informational issues

Further details, including the nature of these issues and recommendations for their remediation, are detailed in the subsequent sections of this report.

3 Summary of Findings

ID	Title	Severity	Status
01	reset_twap Instruction Lacks Verification Associated with Admin Signer	Critical	Fixed
02	smoothing_factor Calculation Loses Precision	Medium	Fixed
03	SPL Stake Epoch Refresh Timeout Check will be Bypassed at the Beginning of Each Epoch	Medium	Fixed
04	SPL Stake Deposit / Withdraw Fee is Not Considered When Calculating Sol Rate	Medium	Fixed
05	SPL Stake Token Price is Incorrect When the Stake Pool is Empty	Low	Acknowledged
06	Decimal of ktokens_token_x is Inconsistent	Low	Fixed
07	update_token_metadata does Not Check the Length of Token Name	Informational	Fixed

4 Key Findings and Recommendations

4.1 reset_twap Instruction Lacks Verification Associated with Admin Signer

Severity: Critical	Status: Fixed
Target: Smart Contract	Category: Data Validation

Description

The anchor accounts for the `reset_twap` instruction are defined in the following code

```
8  pub struct ResetTwap<'info> {
9      pub admin: Signer<'info>,
10     #[account()]
11     pub oracle_prices: AccountLoader<'info, crate::OraclePrices>,
12     #[account(seeds = [b"conf", feed_name.as_bytes()], bump, has_one =
       ↪ admin)]
13     pub configuration: AccountLoader<'info, crate::Configuration>,
14     #[account(mut, has_one = oracle_prices)]
15     pub oracle_twaps: AccountLoader<'info, crate::OracleTwaps>,
```

programs/scope/src/handlers/handler_reset_twap.rs#L8-L16

The admin signer is only bound with the `configuration` account. The `oracle_twaps` and `oracle_prices` accounts are not checked by the conditions in the configuration account macro.

Impact

An attacker could initialize a new configuration account with themselves as the admin. They could then use the admin and configuration accounts they control, and the `oracle_prices and oracle_twaps` accounts from the real scope accounts to reset the twap prices as the current prices.

This seriously undermines the accuracy of TWAP checks, especially for highly volatile tokens.

Recommendation

Add check for `oracle_twaps` account in the configuration account macro.

Mitigation Review Log

Kamino Team: Ok, missing checks added.

Offside Labs: **Fixed**.

Two constraints have been added to the declaration of the Configuration account:

```
has_one = oracle_prices,
has_one = oracle_twaps,
```

4.2 smoothing_factor Calculation Loses Precision

Severity: Medium	Status: Fixed
Target: Smart Contract	Category: Precision

Description

The `get_adjusted_smoothing_factor` function calculates a smoothing factor for EMA based on the time between the last two samples.

However, it was calculated using the u64 type instead of the Decimal type, causing precision loss due to integer rounding. This results in a piecewise function instead of a smooth curve.

```
94            let half_sample_delta = last_sample_delta / 2;
95            let n = (ema_period_s + half_sample_delta) /
                  ↪ last_sample_delta;
96
97            let adjusted_denom = n + 1;
98
99            Ok(Decimal::from(2) / adjusted_denom)
```

programs/scope/src/oracles/twap.rs#L94-L100

Impact

If the delay between two price updates is significant, such as exceeding 40 minutes, the smoothing factor remains constant at 1. This deviates from the expected calculation of the EMA formula.

Proof of Concept

Plot the curve using the following formula corresponding to the code:

```
y = 2 / (1 + floor((3600 + floor(x / 2)) / x)) for x in [30, 3600]
```

Input interpretation

plot	$y = \frac{2}{1+\frac{3600+\frac{x}{2}}{x}}$	x = 30 to 3600

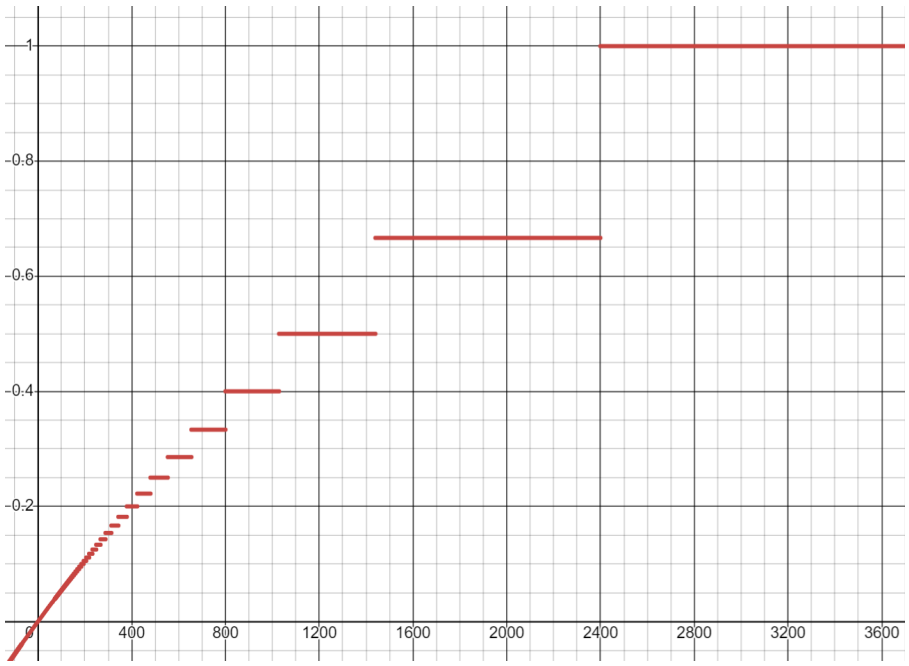

Figure 1: Unsmooth Plot

Recommendation

Also use `decimal_wad::decimal` in the intermediate calculations for `sample_delta`, and simplify the formula to the following:

```
y = 2 / (1 + (3600 / x)) for x in [0, 3600]
```

Input interpretation

plot	$y = \frac{2}{1+\frac{3600}{x}}$	x = 0 to 3600

Figure 2: Smooth Plot

Mitigation Review Log

Offside Labs: Is the CU cost excessively high?

Kamino Team: Ok, formula changed to keep precision

Offside Labs: **Fixed**.

4.3 SPL Stake Epoch Refresh Timeout Check will be Bypassed at the Beginning of Each Epoch

Severity: Medium	Status: Fixed
Target: Smart Contract	Category: Logic

Description

The `spl_stake` oracle includes an update timeout check for `stake_pool.last_update_epoch` ,

```
27          if stake_pool.last_update_epoch != current_clock.epoch &&
              ↪ hours_since_epoch_started >= 1 {
28              // The price has not been refreshed this epoch and it's been
                  ↪ 1 hour
29              msg!("SPL Stake account has not been refreshed in current
                  ↪ epoch");
30              #[cfg(not(feature = "localnet"))]
31              return Err(ScopeError::PriceNotValid.into());
32          }
```

programs/scope/src/oracles/spl_stake.rs#L27-L32

If the `last_update_epoch` is not equal to the current epoch and the current epoch has been underway for over an hour, the price oracle will consider the stake account update as timed out and return a `PriceNotValid` error directly.

The issue is that, if the stake account update remains offline, then for the first hour of each subsequent epoch, the price oracle will update normally. The above check will be bypassed and, furthermore, the price will be updated to the current slot instead of the stake last update slot, `last_updated_slot: current_clock.slot` .

Impact

Expired stake prices, which typically implies significant profits or slashes, will be considered valid. And it will be treated as the valid price for the current slot:

```
43          last_updated_slot: current_clock.slot,
```

programs/scope/src/oracles/spl_stake.rs#L43

Recommendation

```
if stake_pool.last_update_epoch != current_clock.epoch &&
 ↪  (hours_since_epoch_started >= 1
|| stake_pool.last_update_epoch + 1 != current_clock.epoch) {
...
```

Mitigation Review Log

Kamino Team: Ok, fix implemented PR

Offside Labs: **Fixed**.

The issue is a conflict that seems to exist between the elapsed time calculation and this condition.

```
pub fn hours_since_timestamp(current_timestamp: u64, previous_timestamp:
 ↪  u64) -> u64 {
    let seconds_elapsed =
     ↪  current_timestamp.saturating_sub(previous_timestamp);
    seconds_elapsed / SECONDS_IN_AN_HOUR
}

if stake_pool.last_update_epoch != current_clock.epoch &&
 ↪  hours_since_epoch_started >= 1 {
```

If the `stake_pool` has been updated on epoch 1, but it was not updated in the whole epoch 2, and now we are in the first hour of epoch 3, the above condition will be passed. Because `hours_since_timestamp` function uses `current_clock.unix_timestamp - current_clock.epoch_start_timestamp` as the time elapsed.

4.4 SPL Stake Deposit / Withdraw Fee is Not Considered When Calculating Sol Rate

Severity: Medium	Status: Fixed
Target: Smart Contract	Category: Logic

Description

The sol price of spl stake tokens is calculated just as `StakePool.total_lamports / StakePool. pool_token_supply` in the `StakePool::calc_lamports_withdraw_amount` function.

```
247             let numerator = (pool_tokens as
                    u128).checked_mul(self.total_lamports as u128)?;
248             let denominator = self.pool_token_supply as u128;
```

programs/scope/src/oracles/spl_stake.rs#L247-L248

But according to the official document of stake-pool fee best practices. The SPL stake should incur fees both during deposit and withdraw, but this aspect has not been considered in the oracle, leading to higher prices return.

Proof of Concept

In general, deposit fee is zero, but there is also a demonstrative example from the mainnet scope configuration:

The index 72 in the mainnet scope account is bSOL:

```
  "72": {
    "label": "bSOL/SOL",
    "oracle_mapping": "stk9ApL5HeVAwPLr3TLhDXdZS8ptVu7zp6ov8HFDuMi",
    "oracle_type": "SplStake"
  },
```

Dump the fee configuration of bSOL:

```
> spl-stake-pool  list stk9ApL5HeVAwPLr3TLhDXdZS8ptVu7zp6ov8HFDuMi
Stake Pool: stk9ApL5HeVAwPLr3TLhDXdZS8ptVu7zp6ov8HFDuMi
Validator List: 1istpXjy8BM7Vd5vPfA485frrV7SRJhgq5vs3sskWmc
Pool Token Mint: bSo13r4TkiE4KumL71LsHTPpL2euBYLFx6h9HP3piy1
Epoch Fee: 500/10000 of epoch rewards
Stake Withdrawal Fee: 10/10000 of withdrawal amount
SOL Withdrawal Fee: 30/10000 of withdrawal amount
Stake Deposit Fee: 8/10000 of deposit amount
SOL Deposit Fee: 8/10000 of deposit amount
Stake Deposit Referral Fee: 100% of Stake Deposit Fee
SOL Deposit Referral Fee: 100% of SOL Deposit Fee
```

The Deposit Fee of bSOL is 8 basis points, and the Withdrawal Fee is 30 basis points.

Impact

If the deposit/withdrawal fees for stake SOL significantly increase, it could potentially cause a noticeable de-pegging in the current or next epoch. However, the price returned by the Scope cannot reflect this situation. Although the extent of de-pegging in such cases is entirely determined by the market and cannot be accurately assessed, completely ignoring these price differences could potentially lead to malicious arbitrage and result in losses of protocol funds.

Recommendation

It would be better to treat any fee exceeding 2% as a threshold for invalidating the price.

Mitigation Review Log

Kamino Team: We added a check of the range of deposit and withdraw fee so our margins are not invalidated.

Offside Labs: **Fixed**. The fee range check and the method of comparing fees have been well implemented.

4.5 SPL Stake Token Price is Incorrect When the Stake Pool is Empty

Severity: Low	Status: Acknowledged
Target: Smart Contract	Category: Logic

Description

The `StakePool::calc_lamports_withdraw_amount` function calculates the `sol: staked sol spl` . But it returns zero if the `stake_pool.pool_token_supply` is zero.

```
247             let numerator = (pool_tokens as
                ↪ u128).checked_mul(self.total_lamports as u128)?;
248             let denominator = self.pool_token_supply as u128;
249             if numerator < denominator || denominator == 0 {
250                 Some(0)
251             }
```

programs/scope/src/oracles/spl_stake.rs#L247-L250

In fact, it should be 1:1 when the stake pool is empty as the following codes in the `calc_pool_tokens_for_deposit` function

```
124         if self.total_stake_lamports == 0 || self.pool_token_supply == 0 {
125             return Some(stake_lamports);
126         }
```

solana-program-library/stake-pool/program/src/state.rs#L124-L127

Although depositing is the reverse process of withdrawing, if `stake_pool.pool_token_supply` is 0, it is necessary to perform a deposit before withdraw. Therefore, the 1:1 ratio is still correct.

Impact

When the pool is empty, the price returned by the Scope is 0. However, at this point, the price of stake share to asset minted through deposit is actually 1:1. This could potentially lead to instantaneous price manipulation.

Recommendation

Directly returning a 1:1 ratio when `stake_pool.pool_token_supply` is 0 is a valid solution. However, it is crucial to note that introducing an empty SPL stake pool in the oracle price feed is hazardous.

According to the previous audit report on the official Solana SPL stake pool, it was found to be vulnerable to inflation attacks during initialization. Instead of directly fixing this issue, a slippage mechanism was introduced as mitigation.

So, what is truly important is to never include an empty stake pool or a stake pool that can potentially return to be empty in the scope.

Mitigation Review Log

Kamino Team: I see two parts to answer to this remark:

1. We only use already established stake pool so the supply is not 0.
2. In our use case the token's value should be considered nonexistent if there is no supply, which aligns with our current results. Of course, we would need to exercise caution if we were determining the amount of token received when staking SOL.

Offside Labs: Confirmed, this issue indeed falls outside of the considered use cases.

4.6 Decimal of ktokens_token_x is Inconsistent

Severity: Low	Status: Fixed
Target: Smart Contract	Category: Math

Description

The decimals of prices returned by `ktokens_token_x` oracle are inconsistent compared to other oracle feeds. Typically, oracles scale the returned prices based on the decimals precision of the both the numerator and denominator tokens. However, `ktokens_token_x` directly returns the ratio of the quantity of `token_x` to the total shares.

```
132     let num_token_x =
133         holdings_of_token_x(&strategy_account_ref, clmm.as_ref(),
    ↪ &token_prices, token)?;
134     let num_shares = strategy_account_ref.shares_issued;
135     ...
136     let price = u64_div_to_price(num_token_x, num_shares);
```

programs/scope/src/oracles/ktokens_token_x.rs#L132-L139

Impact

If the decimals of `token_x` differs from that of the share token, the price returned by the scope will be scaled by a factor of `10 ** (token_x_decimals - share_decimals)` .This is not currently an issue, as all tokens involved have 6 decimal places.

Recommendation

To ensure consistency, it is recommended to scale the decimal precision of the price in the same manner as other oracles.

Mitigation Review Log

Kamino Team: This was intended at first as the use case was to be able to compute a number of lamports of the "token_x" based on a number of lamports of the "ktoken". This was later changed in order to be able to use USD value in the same context so allowing the use of different price type in the same computation.

Offside Labs: **Fixed**. We have reviewed the scaling algorithm in `price_lamport_to_token_x_per_share` , and it is correct.

4.7 Informational and Undetermined Issues

update_token_metadata does Not Check the Length of Token Name

Severity: Informational	Status: Fixed
Target: Smart Contract	Category: Logic

The ix `update_token_metadata` uses the `Iterator::zip` function to copy the bytes of the input name into the `token_metadata.name` :

```
token_metadata
                .name
                .iter_mut()
                .zip(value.iter())
                .for_each(|(a, b)| *a = *b);
```

The type of `token_metadata.name` is `[u8; 32]`. So if the length of the input name bytes exceeds 32, it will be truncated without any checks or errors, which could lead to misunderstandings about usage and display.

It is recommended to issue an error message if the input name exceeds the maximum length of 32 bytes.

Kamino Team: Added a panic but our cli should prevent us to attempt such update.

5 Disclaimer

This audit report is provided for informational purposes only and is not intended to be used as investment advice. While we strive to thoroughly review and analyze the smart contracts in question, we must clarify that our services do not encompass an exhaustive security examination. Our audit aims to identify potential security vulnerabilities to the best of our ability, but it does not serve as a guarantee that the smart contracts are completely free from security risks.

We expressly disclaim any liability for any losses or damages arising from the use of this report or from any security breaches that may occur in the future. We also recommend that our clients engage in multiple independent audits and establish a public bug bounty program as additional measures to bolster the security of their smart contracts.

It is important to note that the scope of our audit is limited to the areas outlined within our engagement and does not include every possible risk or vulnerability. Continuous security practices, including regular audits and monitoring, are essential for maintaining the security of smart contracts over time.

By using this report, the client acknowledges the inherent limitations of the audit process and agrees that our firm shall not be held liable for any incidents that may occur subsequent to our engagement.

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