sec3™

Security Assessment Report

# Monaco Protocol v0.11.0

July 25, 2023

## Summary

The sec3 team (formerly Soteria) was engaged to do a thorough security analysis of the Monaco Protocol Solana smart contract at https://github.com/MonacoProtocol/protocol. The initial audit was done on the source code of the following version

- **Contract “monaco_protocol”:**
    - v0.11.0, commit 551f1191464a7bf3eae90d92c2ab977f4d12f52c

The review revealed 1 issue, which has been addressed by the team.

# Table of Contents

## Result Overview

In total, the audit team found the following issues.

**MONACO PROTOCOL v0.11.0**

| Issue | Impact | Status |
|---|---|---|
| [ L-1 ] Unsafe type cast | Low | Resolved |

# Findings in Detail

## [ L -1 ] Unsafe type cast

1. The type cast from i128 to u64 at line 49 is not safe. Consider using u64::try_from instead.

```
/* programs/monaco_protocol/src/state/market_position_account.rs */
043 | pub fn total_exposure(&self) -> u64 {
044 |     self.unmatched_exposures
045 |         .iter()
046 |         .zip(&self.market_outcome_sums)
047 |         .map(|(unmatched_exposure, market_outcome_sum)| {
048 |             let postmatch_exposure =
049 |                 (*market_outcome_sum).min(0_i128).checked_neg().unwrap() as u64;
050 |             unmatched_exposure.checked_add(postmatch_exposure).unwrap()
051 |         })
052 |         .max_by(|x, y| x.cmp(y))
053 |         .unwrap()
054 | }
```

2. The type conversion from u64 to i64 at math.rs:29 can be unsafe when it's larger than i64::MAX and smaller than u64::MAX.

```
/* programs/monaco_protocol/src/instructions/math.rs */
028 | pub fn stake_precision_is_within_range(stake: u64, decimal_limit: u8) -> bool {
029 |     Decimal::new(stake as i64, decimal_limit as u32)
030 |         .fract()
031 |         .is_zero()
032 | }
```

3. Similar to 2. However, since this is about time, it's unlikely to go beyond the range. An FYI.

```
/* programs/monaco_protocol/src/instructions/clock.rs */
011 | SystemTime::now()
012 |     .duration_since(UNIX_EPOCH)
013 |     .unwrap()
014 |     .as_secs() as i64
```

4. usize to u32 at payments_queue.rs:63 and market_account.rs:232 can be unsafe on a 64 bit machine, if they are used as an independent utility. However, because the max size of the queue is set to 100, it's safe.

```
/* programs/monaco_protocol/src/state/payments_queue.rs */
044 | impl PaymentQueue {
062 |     pub fn size(&self) -> u32 {
063 |         self.items.len() as u32
064 |     }

/* programs/monaco_protocol/src/state/market_account.rs */
207 | impl Cirque {
231 |     pub fn size(&self) -> u32 {
232 |         self.items.len() as u32
233 |     }
```

## Resolution

The “as 64” in finding #1 was changed to “u64::try_from”. The casting in finding #2 was dropped. For finding #3, an upper bound is set for the testing utility. In finding #4, “usize” was replaced by “u32”.

This issue is resolved.

## Appendix: Methodology and Scope of Work

The sec3 (formerly Soteria) audit team, which consists of Computer Science professors and industrial researchers with extensive experience in Solana smart contract security, program analysis, testing, and formal verification, performed a comprehensive manual code review, software static analysis, and penetration testing.

Assisted by the sec3 Scanner developed in-house, the audit team particularly focused on the following work items:

- Check common security issues.
    - Missing ownership checks
    - Missing signer checks
    - Signed invocation of unverified programs
    - Solana account confusions
    - Arithmetic over- or underflows
    - Numerical precision errors
    - Loss of precision in calculation
    - Insufficient SPL-Token account verification
    - Missing rent exemption assertion
    - Casting truncation
    - Did not follow security best practices
    - Outdated dependencies
    - Redundant code
    - Unsafe Rust code
- 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 the 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 BetDEX Labs (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

Founded by leading academics in the field of software security and senior industrial veterans, sec3 (formerly Soteria) is a leading blockchain security company that currently focuses on Solana programs. We are also building sophisticated 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.

sec3™