UXD Protocol Security Review Public Report

PROJECT: UXD Protocol Security Review

Fall 2021

Prepared For:

UXD Protocol

Prepared By:

Jonathan Haas | Bramah Systems, LLC.

jonathan@bramah.systems

Table of Contents

Executive Summary **3**
Scope of Engagement 3
Engagement Goals 3
Contract Specification 3
Overall Assessment 3
Timeliness of Content 5
General Recommendations **6**
Typographic errors in comments 6
Specific Recommendations **7**
Pubkey Array Contains Containing Uninitialized Members 7
Toolset Warnings **7**
Overview 8
Compilation Warnings 8
Test Coverage 8
Static Analysis Coverage 8
Directory Structure **9**

UXD Protocol Review

Executive Summary

Scope of Engagement

Bramah Systems, LLC was engaged in Fall of 2021 to perform a comprehensive security review of the UXD Protocol smart contracts (specific contracts denoted within the appendix). Our review was conducted over a period of four business days by both members of the Bramah Systems, LLC. executive staff.

Bramah Systems completed the assessment using manual, static and dynamic analysis techniques.

Engagement Goals

The primary scope of the engagement was to evaluate and establish the overall security of the UXD protocol, with a specific focus on trading actions. In specific, the engagement sought to answer the following questions:

- Is it possible for an attacker to steal or freeze tokens?
- Does the Rust code match the specification as provided?
- Is there a way to interfere with the contract mechanisms?
- Are the arithmetic calculations trustworthy?

Contract Specification

Specification was provided in the form of code comments. The contracts were provided via GitHub (commit hash **e71f16f8e5e3f067e7bd55e6c58f8f9b786110f5**).

Overall Assessment

Bramah Systems was engaged to evaluate and identify any potential security concerns within the codebase of the UXD protocol. During the course of our engagement, Bramah Systems found few instances wherein the team deviated materially from established best practices and procedures of secure software development within DLT. UXD has presented a deep and clear design with a detailed focus on security. We found it very easy to follow the defined accounts around, track owners, and easily determine authorization and account confusion vulnerabilities were not possible.

Disclaimer

As of the date of publication, the information provided in this report reflects the presently held, commercially reasonable understanding of Bramah Systems, LLC.'s knowledge of security patterns as they relate to the UXD protocol, with the understanding that distributed ledger technologies ("DLT") remain under frequent and continual development, and resultantly carry with them unknown technical risks and flaws. The scope of the review provided herein is limited solely to items denoted within "Scope of Engagement" and contained within "Directory Structure". The report does NOT cover, review, or opine upon security considerations unique to the Rust compiler, tools used in the development of the protocol, or distributed ledger technologies themselves, or to any other matters not specifically covered in this report.

The contents of this report must NOT be construed as investment advice or advice of any other kind. This report does NOT have any bearing upon the potential economics of the UXD protocol or any other relevant product, service or asset of UXD Protocol or otherwise. This report is not and should not be relied upon by UXD Protocol or any reader of this report as any form of financial, tax, legal, regulatory, or other advice.

To the full extent permissible by applicable law, Bramah Systems, LLC. disclaims all warranties, express or implied. The information in this report is provided "as is" without warranty, representation, or guarantee of any kind, including the accuracy of the information provided. Bramah Systems, LLC. makes no warranties, representations, or guarantees about the UXD protocol. Use of this report and/or any of the information provided herein is at the users sole risk, and Bramah Systems, LLC. hereby disclaims, and each user of this report hereby waives, releases, and holds Bramah Systems, LLC. harmless from, any and all liability, damage, expense, or harm (actual, threatened, or claimed) from such use.

Timeliness of Content

All content within this report is presented only as of the date published or indicated, to the commercially reasonable knowledge of Bramah Systems, LLC. as of such date, and may be superseded by subsequent events or for other reasons. The content contained within this report is subject to change without notice. Bramah Systems, LLC. does not guarantee or warrant the accuracy or timeliness of any of the content contained within this report, whether accessed through digital means or otherwise.

Bramah Systems, LLC. is not responsible for setting individual browser cache settings nor can it ensure any parties beyond those individuals directly listed within this report are receiving the most recent content as reasonably understood by Bramah Systems, LLC. as of the date this report is provided to such individuals.

General Recommendations

Best Practices & Software Development Guidelines

Typographic errors in comments

Numerous code comments contain grammatical errors that can be picked up by a linter. One should be run prior to production deployment.

Examples:

```
       constraint = user_redeemable.amount >= redeemable_amount
@ErrorCode::InsuficientRedeemableAmount
```

Resolution: This has been resolved as of commit hash **79df0f6998f6e941be5abba20b3e6e6773a468e3** through usage of a linter and manual inspection.

Specific Recommendations

Unique to the UXD Protocol

.

Pubkey Array Contains Containing Uninitialized Members

A constraint that was occasionally used was the following:

```
constraint = controller.registered_mango_depositories.contains(&depository.key())
```

At first glance this appears to be sufficient to determine that the specified depository is one that can be used in our request. Pubkeys in the `registered_mango_depositories` array can only be set by previously authorized requests; however, this array, if not full, holds Pubkeys set to the Default value (which is a Pubkey of all zeros). If the specified depository was set to account with a Pubkey matching the default value, this condition would pass despite not being explicitly set.

Of course Anchor does help mitigate this problem. Anchor uses a discriminant to prevent deserializing an Account of an uninitialized or unexpected type. This discriminant uses the first 8 bytes of a SHA256 digest of the namespace (`"account"` by default) and the name of the Account Type (e.g. `MangoDepository`). While incredibly rare, it's possible that the discriminant digest can produce a digest where the first 8 bytes are all 0's.

Anchor, thankfully, protects against further issues when using the `Account<>` type (which validates that this account is owned by the program) but this is not true for `AccountInfo<>`.

Resolution: Anchor's mitigation of this particular issue was deemed sufficient by the team and Bramah.

Toolset Warnings

Unique to the UXD protocol

Overview

In addition to our manual review, our process involves utilizing static analysis and formal methods in order to perform additional verification of the presence of security vulnerabilities (or lack thereof). An additional part of this review phase consists of reviewing any automated unit testing frameworks that exist.

The following sections detail warnings generated by the automated tools and confirmation of false positives where applicable.

Compilation Warnings

No compilation warnings were encountered during the course of our audit.

Test Coverage

The contracts possess a number of functional unit tests encompassing various stages of the application lifecycle.

Static Analysis Coverage

The contract repository underwent heavy scrutiny with multiple static analysis agents, including:

- Semgrep

Directory Structure

At time of review, the directory structure of the UXD Protocol smart contracts repository appeared as it does below. Our review, at request of UXD Protocol, covers the Rust code (*.rs) as of commit hash **e71f16f8e5e3f067e7bd55e6c58f8f9b786110f5**.

```
.
├── Anchor.toml
├── Cargo.lock
├── Cargo.toml
├── README.md
├── package.json
├── programs
│   └── uxd
│       ├── Cargo.toml
│       ├── Xargo.toml
│       └── src
│           ├── error.rs
│           ├── instructions
│           │   ├── deposit_insurance_to_mango_depository.rs
│           │   ├── initialize_controller.rs
│           │   ├── mint_with_mango_depository.rs
│           │   ├── mod.rs
│           │   ├── redeem_from_mango_depository.rs
│           │   ├── register_mango_depository.rs
│           │   ├── set_mango_depositories_redeemable_soft_cap.rs
│           │   ├── set_redeemable_global_supply_cap.rs
│           │   └── withdraw_insurance_from_mango_depository.rs
│           ├── lib.rs
│           ├── mango_program
│          │    ├── anchor_mango.rs
│          │    ├── deposit.rs
│          │    ├── init_mango_account.rs
│          │    ├── mod.rs
│          │    ├── place_perp_order.rs
│          │    └── withdraw.rs
│          ├── state
│          │    ├── controller.rs
│          │    ├── mango_depository.rs
│          │    └── mod.rs
│          └── utils
│               ├── mngo.rs
│               └── mod.rs
├── target
│    └── deploy
│         └── uxd-keypair.json
├── tests
│    ├── identities.ts
│    ├── integration_test_utils.ts
│    ├── oneshot
│    │    ├── mint_wsol.ts
│    │    ├── print_balances_wsol_depository.ts
│    │    ├── rebalance_wsol_depository.ts
│    │    ├── redeem_wsol.ts
│    │    └── wsol_mint_redeem_intensive.ts
│    ├── provider.ts
│    ├── test_0_consts.ts
│    ├── test_0_uxd_api.ts
│    ├── test_1_permissionned_1_setup_controller.ts
│   ├── test_1_permissionned_2_setup_mango_depositories.ts
│   ├── test_1_permissionned_3_set_redeemable_global_supply_cap.ts
│   ├── test_1_permissionned_4_set_mango_depositories_redeemable_soft_cap.ts
│   ├── test_1_permissionned_5_deposit_insurance_on_mango_depository_wsol.ts
│   ├── test_1_permissionned_6_withdraw_insurance_from_mango_depository_wsol.ts
│   ├── test_2_consts.ts
│   ├── test_2_mango_depository_1_btc.ts
│   ├── test_2_mango_depository_2_wsol_1.ts
│   ├── test_2_mango_depository_3_wsol_test_redeemable_global_cap.ts
│   └── test_2_mango_depository_4_wsol_test_redeemable_soft_cap.ts
├── tsconfig.json
├── uxd.jpg
└── yarn.lock

11 directories, 54 files
```